UNIVERSIDAD NACIONAL FEDERICO VILLARREAL

ESCUELA UNIVERSITARIA DE POST GRADO

Base de datos distribuidos usando algoritmos genéticos para optimización de proceso transacción en la Web

TESIS:

para optar el grado académico de Doctora en Ingeniería

AUTOR:

Luzmila Elisa Pró Concepción

ASESOR:

Doctor César Armando Zarate Gonzáles

Lima – Perú

2010

RESUMEN

BASE DE DATOS DISTRIBUIDOS USANDO ALGORITMOS GENÉTICOS PARA OPTIMIZACIÓN DE PROCESO TRANSACCIÓN EN LA WEB

POR

LUZMILA ELISA PRÓ CONCEPCIÓN

AGOSTO -2010

Asesor : Doctor Ph César Armando Zarate Gonzáles

Tesis para optar el Grado Académico de Doctora en Ingeniería.

EL desarrollo de la investigación de la Tesis de "Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la *Web*", ha sido posible llegar a las siguientes conclusiones:

Hay deficiencia en el tiempo de procesos de transacción por el procesador del servidor; que actualmente trabajan con algoritmos tradicionales; como la lectura / escritura de datos en el disco magnético en el servidor Web, produciéndose por ejemplo, demora en la cola de espera, demora en tiempo de proceso de transacción, demora en tiempo de respuesta, que ocasionan los denominados cuellos de botella, falta memoria, etcétera.

El problema central que se propone está orientado al crecimiento y, evolución del servidor web de una manera económica y escalable que lleva a un rendimiento óptimo. Por consiguiente, existe la necesidad de estudiar los procesos de transacciones del sistema, de tal manera que se aplique otra alternativa como algoritmos genéticos para optimizar el proceso de transacción en el servidor, a fin de así mejorar los procesos del servidor web y, mejorar la atención a los clientes / usuarios.

El objetivo es implementar un simulador de transacciones orientando a la toma de decisiones del administrador de transacciones con la aplicación de algoritmos genéticos. Se usará los algoritmos genéticos para determinar que transacción se debe tomar para asignarlo en la cola de procesos.

Se asumen ciertas restricciones que el simulador tomará como dadas. Por ejemplo, cada transacción tiene un número constante de recursos que solicitan. Cada recurso tiene una cola que administra y solo se pueden hacer 2 tipos de requerimiento: leer y escribir.

La estructura de un cromosoma consta de un grupo de alelos y cada uno corresponde con un recurso solicitado. El administrador de transacciones tomará el

requerimiento por el recurso para ponerlo en cola, el que tenga un máximo de cromosoma igual a 1, es decir, cuando encuentre entre el grupo de transacciones la transacción que tenga sus alelos en 1. Se tomará como cromosoma una cadena binaria que será convertida a números enteros.

Se ha realizado un análisis de los modelos de transacciones que operan actualmente y se ha extraído tales mecanismos para llevarlo a un proceso de toma de decisiones en función de los algoritmos genéticos.

Se ha implementado un simulador prototipo para un sistema de aplicación con algoritmos genéticos, para optimizar el proceso de transacción, antes de procesar los datos, se evaluarán los procesos de transacciones sobre: tiempo de simulación, número de transacciones, tiempo de la transacción, número de recursos, longitud de la cola del recurso, probabilidad de cruzamiento, probabilidad de mutación, transacciones en cola, atendidos, en lectura, en escritura, tiempo consumido, y se consigue los resultados de procesos óptimos; el tiempo de procesamiento de datos mediante el simulador es menor que el tiempo de procesamiento de datos que en el procesador convencional, mejor uso del recurso de la computadora.

PALABRAS CLAVES: ALGORITMOS GENÉTICOS.
BASE DE DATOS DISTRIBUIDOS.
GENOMA.
INTERCONEXIÓN DE SISTEMAS ABIERTOS.
PROTOCOLO DE CONTROL DE TRANSMISIÓN.
PROTOCOLO DE INTERNET.
SERVIDOR WEB.
SISTEMA OPERATIVO.
TRANSACCIÓN.

ABSTRACT

BASE OF DISTRIBUTED DATA USING GENETIC ALGORITHMS FOR OPTIMIZATION OF PROCESS TRANSACTION IN THE WEB

FOR

LUZMILA ELISA PRÓ CONCEPCIÓN

AUGUST-2010

Advisory : Doctor Ph César Armando Zarate Gonzáles

Thesis to opt Doctor's Academic Grade in Engineering.

The development of the investigation of the Thesis of "Distributed Database Using Genetic Algorithms For Optimization of Process Transaction in the Web" that have been allowed to reach the following conclusions:

There is deficiency in the time of transaction processes for processor of the server; that they are working with traditional algorithms; as the reading / writing of data in the magnetic disk in server web. For example, it delays in the wait line, it delays in time of transaction process, it delays in time of answer that you/they cause as neck of the bottle, it lacks memory, etc.

The central problem that intends is guided to the growth and, evolution of the server web in an economic and scalable way that takes to a good yield. Consequently, the necessity exists of studying the processes transactions of the system, in such a way that another alternative is applied as genetic algorithms to optimize the process of transaction in servant, basing stops to improve processes of the server web and, to improve the attention to the clients / users.

The objective is to implement a pretender of transactions guiding the taking of the administrator's of transactions decisions with the application of genetic algorithms. It was used the genetic algorithms to determine that transaction should take to assign it in the line of processes.

Certain restrictions are assumed that the pretender took as given. For example, each transaction has a constant number of resources that you/they request. Each resource has a line that he/she administers and alone 2 requirement types can be made: to read and to write.

The structure of a chromosome consists of an alelos group and each one corresponds with a requested resource. The administrator of transactions took the requirement for the resource to put it in line, the one that has a maximum of chromosome similar to 1, that is to say, when he finds among the group of transactions the transaction that has his alelos in 1. He took as chromosome a binary chain that will be converted to whole numbers.

An analysis of the models of transactions has been made that they operate at the moment and it has been extracted such mechanisms to take it to a process of taking of decisions in function of the genetic algorithms.

Pretender prototype has been implemented for system application with genetic algorithms, to optimize the process of transaction, before to process the data processes of transactions will be evaluated of: time of simulation, number of transactions, time of the transaction, number of resources, longitude of the line of the resource, crossover probability, mutation probability, in line, assisted, reading, notarizes, consumed time, and it is gotten the results of good processes; the time of prosecution of data is less than the time of prosecution of data that conventional in the processor, better use of the resource of the computer.

KEY WORDS: GENETICS ALGORITHMS.
BASE OF DISTRIBUTED DATA.
GENOMA.
INTERCONNECTION OF OPEN SYSTEMS.
PROTOCOL OF CONTROL OF TRANSMISSION.
PROTOCOL DE INTERNET.
SERVER WEB.
OPERATING SYSTEM.
TRANSACTION.

JURADOS DE LA TESIS DE DOCTORA EN INGENIERÍA

LUZMILA ELISA PRÓ CONCEPCIÓN

Tesis Doctoral presentada a consideración del Cuerpo Docente de la Escuela Universitaria de Post Grado, de la Universidad Nacional Federico Villarreal, como parte de los requisitos para obtener el Grado Académico de Doctora en Ingeniería:

Aprobado por:

_____
Dr. Walter Esteban Barrutia Feijoo
Presidente

_____
Dr. Abimael Arsenio Guzmán Jorquera
Miembro

_____
Dr. Justo Pastor Solis Fonseca
Miembro

_____
Dr. Ph César Armando Zarate Gonzáles
Asesor de la Tesis

Lima - Perú

2010

FICHA CATALOGRÁFICA

PRÓ CONCEPCIÓN LUZMILA ELISA

"Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web", (Lima) 2010.

Xii, 154 páginas, 21.59-29.7 cm, (UNFV, Doctora en Ingeniería, 2010).

Tesis, Universidad Nacional Federico Villarreal,
Escuela Universitaria Post Grado 1
I. UNFV / EUPG II. Grado (Serie).

AGRADECIMIENTOS

Mi sincero agradecimiento al Dr. Ph César Armando Zarate Gonzáles, mi asesor de la Tesis Doctoral, por su apoyo continuo y ayuda en asesoría del tema. Por haber tenido paciencia y cooperación por toda investigación, sin ello, no habría sido posible lleve a cabo la disertación.

Mis agradecimientos a los señores doctores Walter Esteban Barrutia Feijóo, Oscar Hugo Mujica Ruiz, Abimael Arsenio Guzmán Jorquera, Justo Pastor Solis Fonseca, para cada uno ellos, por haber dedicado su tiempo en la revisión y asesoría del Tema de Tesis, Plan de Tesis y la Tesis Doctoral de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso de Transacción en la Web”.

Mis agradecimientos finales a todas las personas que han colaborado en la elaboración de la presente tesis doctoral con su apoyo directo o indirecto a la investigación.

Luzmila Elisa Pró Concepción

ÍNDICE

ÍNDICE

Página

ÍNDICE

CAPÍTULO V

CAPÍTULO VI

ÍNDICE

CAPÍTULO VII

ÍNDICE

Página

LISTA DE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

CAPÍTULO V

CAPÍTULO VI

CAPÍTULO VII

ÍNDICE

Página

LISTA DE TABLAS

INTRODUCCIÓN

La presente disertación trata de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web”, se le ha identificado dentro de la taxonomía de ingeniería de software en la web, para el desarrollo del estudio se ha requerido la técnica de caché, proceso de transacción, asimismo, un conjunto de procedimientos para el proceso de transacción en el servidor Web y filtrándose sobre un período tiempo prolongado y base de datos distribuidos en el servidor Web de diferentes fuentes.

La tecnología de **Base de Datos Distribuidos** (BDD), es la unión de dos ideas ligadas al procesamiento de datos: **Sistema Gestión de Base de Datos** (DBMS) y procesamiento distribuido de datos sustentado en el empleo de redes de computadoras.

Una de las motivaciones del uso de **Base de Datos** (BD), es la necesidad de integrar los datos operacionales de una organización y proveer una centralización que controle el acceso a los mismos. La tecnología de redes de computadoras, por otro lado, promueve un modo de trabajo que procura, a grandes rasgos, evitar la centralización. Estas dos tendencias, a priori contrapuestas, tienen en conjunto el objetivo de establecer una tecnología en la base de datos que apunte a la integración de la información, más que a la centralización de la misma.

En el presente estudio de investigación de Tesis Doctoral, se ha identificado procesos de tareas en el servidor, de lo cual se puede decir que hay deficiencia en el tiempo de procesos de transacciones en el sistema; que estan trabajando con algoritmos tradicionales, como la lectura / escritura de datos en el disco magnético en el servidor web, por ejemplo, demora en tiempo de espera para el proceso de transacción, demora en tiempo de respuesta, que ocasionan los denominados cuellos de botella, y falta memoria.

Se elaborará la metodología nueva para el sistema de aplicación usando algoritmos genéticos, para optimizar eventos concurrentes de proceso de transacción en el servidor Web, que permitirá ahorro de: tiempo de espera de las colas, tiempo de proceso de tareas, menor tiempo de lectura / escritura datos en disco magnético del servidor web de la empresa o institución. Para está investigación, se utilizarán **Protocolo de Internet / Protocolo de Control Transmisión** (TCP/IP) para la transmisión de datos (paquetes) en las redes de computadoras.

El **Capítulo I** trata de “Teoría base”, que comprende un conjunto de conocimientos relacionados como: antecedentes de la base de datos, disco duro (transacción), escalabilidad, Internet, sistema de comunicaciones.

El **Capítulo II** trata de “Algoritmos Genéticos”, considerado como una técnica de búsqueda aleatoria dirigida, que puede encontrar la solución óptima global en los espacios de búsqueda multidimensionales complejos. Los Algoritmos Genéticos, es

un modelo de la evolución natural que los operadores emplean y que está inspirado por el proceso de la evolución natural, lo cual permitirá aplicar a los procesos de transacciones en el servidor.

El **Capítulo III** trata de “Sistemas Distribuidos”, que puede ser considerado como una colección de elementos de cómputo autónomo que se encuentran físicamente separados y no comparten una memoria común, se comunican entre sí a través del intercambio de mensajes utilizando un medio de comunicación. Los sistemas autónomos, pueden tener características homogéneas o heterogéneas

El **Capítulo IV** trata de “Base de Datos Distribuidos”, considerándolo a la base de datos distribuidos, como una unidad virtual, cuyas partes se almacenan físicamente en varias bases de datos "reales" distintos, ubicados en diferentes sitios.

A diferencia de los sistemas paralelos, en los que los procesadores se hallan estrechamente acoplados y constituyen un solo sistema de bases de datos, los sistemas de bases de datos distribuidos, están formados por sitios débilmente acoplados que no comparten ningún componente físico. Además, dado que, los sistemas de bases de datos que se ejecutan en cada sitio pueden ser sustancialmente independientes entre sí.

El **Capítulo V** trata de “Gestión de Transacción”, una transacción es una unidad de ejecución de un programa que accede y posiblemente actualiza varios elementos de datos. Una transacción, se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación (por ejemplo SQL, C++ o Java), y está delimitado por instrucciones (o llamadas a función) de la forma **inicio transacción** (begin transaction) y **fin transacción** (end transaction). La transacción, consiste de todas las operaciones que se ejecutan entre begin transaction y end transaction (Silberschatz, Korth, Sudarshan 2006).

El **Capítulo VI** trata de “Metodología de implementación del simulador prototipo de aplicación utilizando algoritmos genéticos para optimizar procesos de transacción en la web”, que nos permitirá construir un simulador prototipo de base de datos distribuidos usando algoritmos genéticos para optimización de proceso de transacción en el servidor web.

El **Capítulo VII** trata de “Análisis e interpretación de resultados”, se realiza la prueba del simulador prototipo y se analizarán los resultados obtenidos del trabajo de investigación de la tesis doctoral.

a) PLANTEAMIENTO DEL PROBLEMA

Hoy en día, en los procesos de transacción de base de datos distribuidos hay deficiencias en los procesos de transacciones; como la lectura / escritura de datos en el disco magnético en el servidor web. Por ejemplo, demora en el tiempo de espera de proceso, demora en tiempo de respuesta, que ocasionan los denominados cuellos de botella, y falta memoria por tiempo de espera en la CPU.

El problema central, que se propondrá está orientado al crecimiento proliferado de Internet; de una manera económica y escalable que lleve a un rendimiento óptimo. Por consiguiente, existe la necesidad de optimizar los procesos de transacciones en las bases de datos distribuidos en el servidor web, de tal manera, que se aplique otra alternativa como son los algoritmos genéticos para optimizar los procesos, a fin de mejorar los procesos distribuidos del servidor de base datos en la web y, para obtener calidad de servicio para clientes / usuarios.

i) ALTERNATIVAS DE SOLUCIONES DEL PROBLEMA PROPUESTA

Hay dos alternativas de soluciones de:

1. Redes neuronales.
2. Algoritmos genéticos.

Se ha elegido la alternativa de los algoritmos genéticos, para solucionar el problema propuesto. Los Algoritmos genéticos, que por su evolución natural son justamente utilizados en optimización, en este caso, se tratan de optimizar los procesos de transacciones en el procesador del servidor Web. Se utilizarán los algoritmos genéticos para desarrollar el simulador prototipo para optimizar los procesos de transacción en el servidor Web, más adelante se detallan los algoritmos genéticos.

b) JUSTIFICACIÓN E IMPORTANCIA

La presente Tesis Doctoral, Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web, se justifica su estudio porque optimizará los procesos de transacciones en las bases de datos distribuidos en el servidor Web, mejorando el procesamiento de datos optimizando los tiempos utilizados en procesos de transacción de tareas de eventos concurrentes en el servidor y las computadoras de las estaciones de trabajo.

La implementación del simulador con algoritmos genéticos, permitirá: ahorro de: tiempo de espera de las colas, tiempo de proceso de transacción, menor tiempo de lectura / escritura datos en disco magnético del servidor web de la empresa o institución. Para está investigación se utilizarán **Protocolo de Internet / Protocolo de Control Transmisión** (TCP/IP), para la transmisión de datos (paquetes) en las redes de computadoras, entonces, a fin de mejorar el tiempo de proceso de transacción en el servidor, la calidad de servicio para los usuarios / clientes, etcétera.

c) OBJETIVOS

Desarrollar un simulador prototipo usando algoritmos genéticos para optimización de procesos de transacciones en el servidor *web*, a fin de mejorar el rendimiento en la lectura / escritura de los datos en la **Unidad Central de Procesamiento** (CPU) y que es atendido por el despachador del servidor.

OBJETIVOS ESPECÍFICOS

- Analizar los procesos de transacciones de clientes / usuarios en el servidor.
- Aplicar por primera vez algoritmos genéticos para procesos de transacciones.
- Implementar un simulador prototipo utilizando algoritmos genéticos para optimizar procesos de transacciones en el servidor web.
- Analizar los resultados de procesos de transacciones de datos.
- Disminuir el tiempo de espera de las colas en el servidor.
- Disminuir el tiempo de demora de procesos de transacciones en la computadora.
- Disminuir los costos de recursos computacionales del servidor.
- Mejorar el tiempo de respuesta de mensajes del sistema.
- Permitir que el simulador prototipo propuesto con algoritmos genéticos mejorará los tiempos utilizados para procesos de transacciones del procesador.

d) HIPÓTESIS

El sistema de aplicación con algoritmos genéticos en el proceso transacción del servidor mejorará los tiempos utilizados de proceso de transacción del servidor y mejorará la calidad de servicios para atender a los usuarios / clientes.

De la Tesis Doctoral de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web”, se considerará las variables siguientes:

i) VARIABLE INDEPENDIENTE

Simulador por computadora que utiliza algoritmos genéticos para optimizar proceso de transacción en la web, como rendimiento, carga, recursos, ancho de banda, tiempo de respuesta, velocidad de transferencia.

ii) VARIABLE INTERDEPENDIENTE

Algoritmos genéticos para optimización de procesos de transacciones en el servidor web.

iii) VARIABLE DEPENDIENTE

Optimización de procesos de transacciones concurrentes en el servidor web.

e) METODOLOGÍA

La metodología de investigación es de tipo experimental, es un modelo aplicado, que ha sido definido para guiar paso a paso el desarrollo de la Tesis Doctoral, de acuerdo a una investigación científica.

La presente metodología de investigación experimental incluye el modelo cliente / servidor utilizando lenguaje de programación C ++ y algoritmos genéticos, para desarrollar el simulador prototipo de aplicación usando algoritmos genéticos para proceso de transacción de eventos concurrentes, en base de datos distribuidos de la tesis propuesta, a continuación algunas consideraciones:

1. Se ha elegido la metodología experimental de base de datos distribuidos de proceso de transacción en el servidor Web con tecnología Internet; que permitirá desarrollar la investigación de la Tesis Doctoral de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web”, considerando la secuencia de pasos que se van a seguir y detalles correspondientes al tema.

2. La metodología de algoritmos genéticos, es la base principal para implementar el simulador prototipo de aplicación del sistema de optimización basado en algoritmos genéticos para proceso de transacción en el servidor.

3. La población, es un conjunto de individuos del genoma de algoritmos genéticos, en el sistema, se toman los cromosomas fuertes y (los débiles se rechazan) para la optimización de procesos de transacciones en la base de datos distribuidos.

4. La implementación del simulador prototipo de aplicación usando algoritmos genéticos para optimización de proceso de transacción en el servidor, será probada en el Laboratorio de Cómputo de la Facultad de Ingeniería de Sistemas e Informática de la UNMSM.

CAPÍTULO I

1. TEORÍA BASE

1.1 ANTECEDENTES

Los sistemas de base de datos, tuvieron sus orígenes en (1960-1962), cuando se empezaron a utilizar las máquinas que codificaban la información en tarjetas perforadas por medio de perforaciones. Las bases de datos se crean con el objetivo de almacenar grandes cantidades de datos, antes se almacenaban en libros, lo que era lento, costoso y complejo; realizar cualquier actualización, había que hacerla en cada una nuevamente en que apareciera dicha información a modificar.

DISCO DURO. Esta compuesta por varios platos, es decir, varios discos de material magnético montados sobre un eje central sobre el que se mueven. Para leer y escribir datos en estos platos, se utilizan las cabezas de lectura / escritura que mediante un proceso electromagnético codifican / decodifican la información que han de leer o escribir. La cabeza de lectura / escritura en un disco duro, está muy cerca de la superficie, de forma que casi da vuelta sobre ella, sobre el colchón de aire formado por su propio movimiento. Debido a esto, están cerrados herméticamente, porque cualquier partícula de polvo puede dañarlos. Este se divide, en unos círculos concéntricos cilíndricos (de similar organización con las pistas de los disquetes), que empiezan en la parte exterior del disco (primer cilindro) y terminan en la parte interior (último). Asimismo, estos cilindros se dividen en sectores, cuyo número esta determinado por el tipo de disco y su formato, siendo todos ellos de un tamaño fijo en cualquier disco. Cilindros como sectores se identifican con una serie de números que se les asigna, empezando por el 1, pues el número 0 de cada cilindro se reserva para propósitos de identificación más que para almacenamientos de datos. Estos datos escritos / leídos en el disco, deben ajustarse al tamaño fijado del almacenamiento de los sectores. Habitualmente, los sistemas de discos duros contienen más de una unidad en su interior, por lo que el número de caras puede ser más de dos. Estas se identifican con un número, siendo el 0 para la primera. En general, su organización es igual a los disquetes. La capacidad del disco resulta de multiplicar el número de caras de pistas por cara y por sectores por pista, al total por el número de octetos por sector.

A partir del año de 1970, surgieron las primeras minicomputadoras, que competirían con las grandes computadoras, tanto por las prestaciones, como por su precio, con lo que se extendió su uso. Los grandes sistemas centralizados, fueron dejando paso lentamente a sistemas mucho más descentralizados, y formados por varias computadoras o a sistemas multiprocesadores. Pronto surgieron nuevas necesidades de interconexión entre los equipos, como computadoras personales y también utilizan para estaciones de trabajo en la red, y se desarrollaron las **redes de área local** (LAN), como Ethernet o Token ring, etcétera.

El uso de sistemas de base de datos automatizados, se ha desarrollado a partir de la necesidad de almacenar grandes cantidades de datos, para su posterior consulta, producidas por las nuevas industrias que creaban gran cantidad de información.

Tal vez, uno de los aspectos que se ha notado más recientemente en el campo de las bases de datos (como en casi cualquier otro campo de la informática), es el crecimiento vertiginoso de la WWW en la Internet. La conexión de las bases de datos con la web ha ido progresando desde una interrelación realizada a través de herramientas ad hoc hasta la situación actual, en la que prácticamente todo SGBD proporciona un módulo o toda una serie de herramientas para publicar la información de la base de datos en la red, siendo accesible desde cualquier punto, utilizando un navegador (Almquist 1992). Con el uso de Internet y de las intranets, la disponibilidad de los datos de la organización se ha hecho prácticamente ubicua, sin necesidad de operaciones más comunes y sencillas del desarrollo de ninguna aplicación cliente, exceptuando el navegador. Los catálogos, inventarios, stocks, indicadores, etcétera, de cualquier empresa o comercio están disponibles a cualquier cliente / usuario en cualquier momento, con sus respectivos permisos y de manera concurrente.

Es de destacar, que la tecnología Web, ha hecho evolucionar la tecnología cliente / servidor de dos capas a una tecnología comúnmente estructura en tres capas (1. Cliente, 2. Servidor de aplicaciones, 3. Servidor de datos), aunque la mayoría de los aspectos del paradigma cliente / servidor son aplicables a la web.

En definitiva, con el advenimiento de la web, la gestión de datos se ha ramificado para tratar con la variedad de información disponible en la **telaraña alrededor del mundo** (WWW). La mayoría de los accesos web actual disparan alguna forma de generación de contenido de una base de datos, mientras que el comercio electrónico, está destinado a hacer un uso intensivo de las aplicaciones basadas en un **Sistema de Gestión de Base de Datos** (SGBD).

En consecuencia de todo esto, es el interés de la comunidad científica y de las empresas / instituciones del sector que se centran en la revisión o extensión de modelos de datos y de lenguajes de consulta, la integración de datos tan diversos, la reconcepción de los índices, las transacciones y procesamiento de consultas, con objetivo de adaptarse a las características y a la escala de los datos en la web. Se han identificado nuevos problemas, como saber tratar con el solapamiento de información y la detección de copia, así como cuestiones específicas de la web como herramienta de publicación. También, hay un gran interés por el **lenguaje de marcas hipertexto** (HTML), por la extracción y recogida de información de la web, su almacenamiento en un almacén de datos y su prospección.

1.2 ESCALABILIDAD

La escalabilidad puede definirse como la capacidad de aumentar el tamaño del dominio del problema con un aumento pequeño o no significativo de tiempo de solución y complejidad del espacio. La escalabilidad de la arquitectura microprocesador del servidor Web, es la capacidad de aumentar el número de servidores, clientes / usuarios, tipo de datos, tamaño de datos y la capacidad de manejo de servidores ampliamente en ubicación de cobertura geográfica, con un mínimo cambio en la calidad de servicio. En este capítulo, se analizarán los métodos y técnicas que ayudarán a escalar en la web. Se iniciará identificando técnicas que mejoran la escalabilidad; de otra manera las soluciones de ancho de banda de la red creciente y rendimiento específico del servidor. Estas técnicas son: el uso del caché del servidor; por ejemplo, (Kwan, McGrath y Reed 1995), afirmaban que con el usuario y con el caché se construye navegadores web, y conocidos como caché del servidor Proxy, y creando clientes mejores en la web, mejor protocolo, compresión, y finalmente se logra que la red sea escalable.

1.2.1 UTILIZACIÓN DE CACHÉ

El caché se puede implementar en tres lugares: en servidores, en la red y en los usuarios. El caché implementado en servidor, refleja el interés global del contenido del servidor. Puede ser implementado reproduciendo el sistema del archivo y el servidor de HTTP y conectando a los servidores reproductores con una red de alta velocidad (Kwan, McGrath, Reed 1995). Esto es similar al servidor reflejado, de donde los datos del servidor se copian a otros servidores para reducir la carga en la red y del servidor original. Sin embargo, en el servidor que refleja a los servidores no se ponen en su ubicación; ellos pueden separarse por distancias lejanas o remotas. El servidor de caché, se utiliza principalmente para reducir la carga en servidor y mejorar el tiempo de respuesta del rendimiento total.

El caché de cliente, refleja los intereses del usuario. El contenido de caché cambiará según la especificación de acceso de usuario y el tamaño del caché. El caché de red, funcionará según la especificación de acceso de un grupo de usuarios que comparten de caché. La efectividad de la red de caché, puede aumentar poniéndolo donde se sabe que un grupo de usuarios tiene un alto grado de similitud, interés o situación de referencia y así, se da la implementación múltiple (Malpani, Lorch, Berger 1995) o caché jerárquico del servidor Proxy. Existe el caché múltiple del servidor Proxy, donde muchos usuarios que comparten cachés y un caché extra pueden consultar a otros cachés. El caché de nivel dos, es aquel que se tiene en varias redes conectados con otro caché del servidor web o caché de red de primer nivel, y con tamaño grande. Si el documento no se encuentra en el caché de primer nivel, entonces, accesará a un caché de segundo nivel, en donde se encontrará el documento buscado.

El caché construye almacenamiento dinámico de datos para los usuarios, no provee una solución mágica para el problema de la escalabilidad, sin embargo, ayuda a resolverlo. Combinando con otras técnicas, como la compresión de datos y la repetición del servidor, se podría tener mejores redes para mantener la arquitectura web que sea utilizable en el futuro.

1.2.2 PROTOCOLO DE TRANSFERENCIA DE HIPERTEXTO MEJORADO

El **Protocolo de Transferencia de Hipertexto** (HTTP), soporta varios problemas; no ha sido diseñado para optimizar la interacción (Gettys 1995). Un problema con el protocolo original, es el uso de una nueva conexión para cada documento recuperado de la red. Este crea los paquetes extras sobre la red y causa la congestión de la red.

El Protocolo HTTP de versión mejorada, es un modelo de distribución y asignamiento de direcciones de cada usuario / cliente entre las computadoras personales o redes de computadoras, que permitirá dar soluciones de administración de ubicación de los usuarios / clientes; siempre que un documento es accesado por un usuario que extrae del servidor original. Esto aumenta la carga en los servidores, y el tráfico de red se incrementa. Reflejando el uso de servidores web, se hace actualmente creando manualmente una copia de documentos en otro servidor y con otra ubicación. También, se utiliza caché en el servidor Proxy, para documentos accesados; sin embargo, hay todavía, una necesidad de notificar espejos o cambios de documentos en caché.

Un ejemplo específico es la entrega de vídeo y audio en la red. El método actual, es utilizado por otro protocolo de transacciones. Esto introduce el arranque elevado de un protocolo nuevo de comunicación. El protocolo HTTP, podría diseñarse para utilizar las funcionalidades disponibles en los protocolos, para la entrega más rápida y mejora de trabajo del protocolo HTTP.

1.2.3 COMPRESIÓN DE DATOS

Es la técnica que reduce el tamaño de los archivos, tal que, una unidad de almacenamiento contenga la mayor cantidad de datos, lo cual permitirá el envío de datos a través de las conexiones, reduciendo eficientemente, reduciendo el número de bits que se envían por un “enlace de datos” para representar mensajes. Estas técnicas, están basadas generalmente en esquemas para reducir el número de caracteres que se repiten o representando con un número variable de bits, dependiendo, de cuántas veces es enviado cierto carácter. El proceso de compresión de archivos se realiza por medio de la eliminación de datos redundantes y no críticos, de manera que pueda almacenarse y transmitirse con mayor eficiencia. Los trabajos de compresión, adecuados con documentos del texto, sin embargo, pueden ser menos eficaces con los archivos binarios, sobre todo si ellos ya están comprimidos como JPEG o GIF, etcétera. Utilizando el porcentaje de archivos comprimibles y el promedio de entropía

en cada uno de esos tipos de archivos, se puede estimar cuánto de ancho de banda se necesita y ahorrar a través de la compresión.

1.3 INTERNET

1.3.1 EVOLUCIÓN DE LA INTERNET

La evolución de Internet, empezó a finales de 1960. La Unión Soviética, había lanzado el satélite Sputnik en 1957, y en plena guerra fría los Estados Unidos de América, querían asegurarse su posición a la cabeza de la tecnología militar. Entonces, el **Departamento de Defensa de los Estados Unidos** (DoD), se dió cuenta de que la tecnología empleada por la red telefónica, llamada conmutación de circuitos, era demasiado frágil para resistir el más mínimo ataque y mucho menos la tan temida guerra nuclear. Si se destruía una conexión entre dos centrales importantes o resultaba una central fuera de servicio, alguna parte de las telecomunicaciones de defensa del país podrían quedar inutilizadas.

Posteriormente, como en el caso de muchas otras tecnologías, Internet y las redes de conmutación de paquetes, se desarrollaron inicialmente gracias al financiamiento y apoyo del gobierno de Estados Unidos de América. La **Oficina de Proyectos de Investigación Avanzada** (ARPA) de los Estados Unidos, fue una de las primeras instituciones que adoptó la teoría de conmutación de paquetes. ARPA, creó lo que fue llamado ARPAnet, como una red importante de computadoras del gobierno capaces de resistir daños a la red producidos por una guerra o una catástrofe severa. Con la colaboración de varias compañías y universidades, los esfuerzos iniciales de ARPA, culminaron en septiembre de 1969, cuando fue entregada a la **Universidad de California en Los Ángeles** (UCLA), una minicomputadora Honeywell 516. Este sistema fue el primero de cuatro conmutadores de paquetes, también, conocido como **procesadores de mensajes con interfases** (IMP). Otros tres conmutadores de paquetes fueron instalados en la **Universidad de Utah**, la **Universidad de California en Santa Bárbara** y el **Instituto de Investigación de Stanford**. Muy pronto, estas computadoras intercambiaban paquetes entre sí a través de líneas telefónicas, con lo que nació la “madre de Internet”: ARPAnet. Se muestra en la figura 1.1 los cuatro sitios originales de ARPAnet (Servati, Bremmer, Lasi 1998).

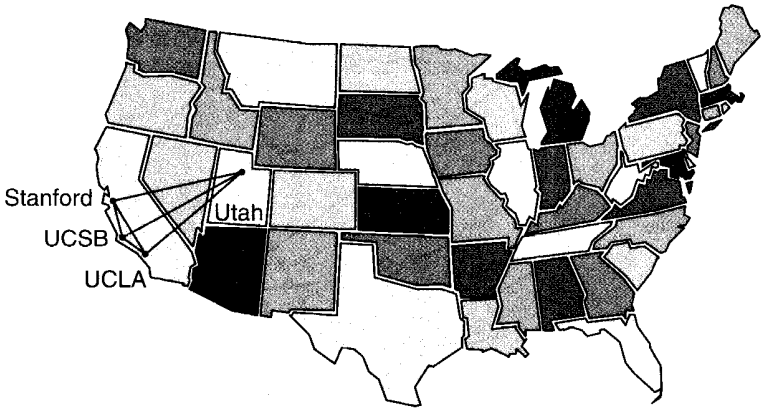


Figura 1.1. La ARPAnet original.

Hoy en día, la tecnología de Internet, esta sofisticado de *hardware* y *software* corporativos en el ámbito mundial, proliferación de: clientes, mensajes, comunicaciones, etcétera, en general, se muestra en la figura 1.2 siguiente:

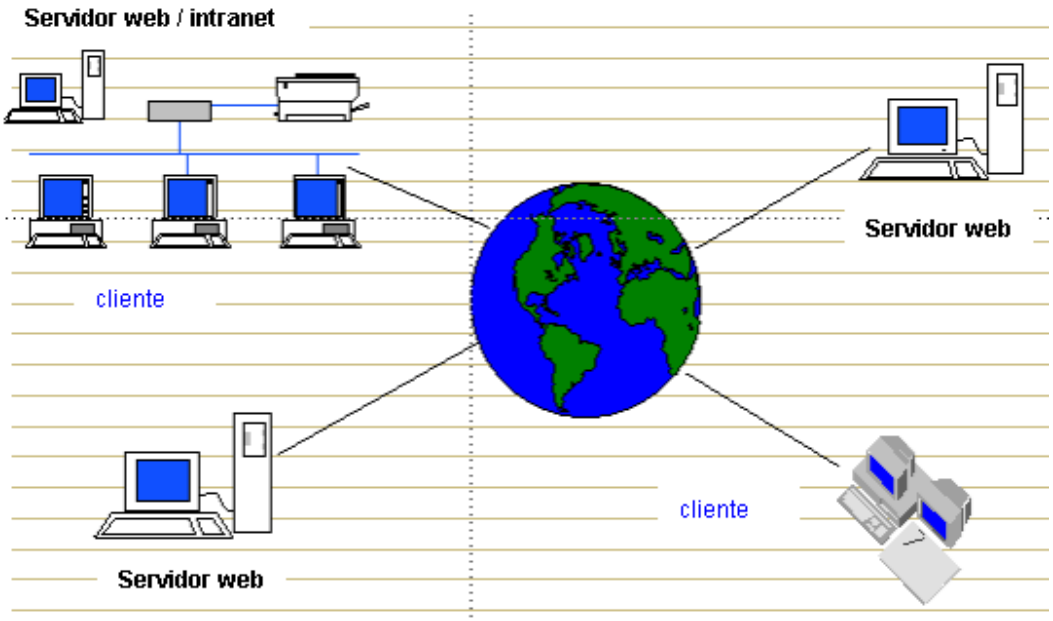


Figura 1.2. Redes de computadoras en la Internet.

1.3.1.1 DESARROLLO EN INTERNET2

Los objetivos mencionados anteriormente, son llevados a cabo mediante actividades de desarrollo y prueba de nuevos protocolos y aplicaciones para Internet2. Estos desarrollos son hechos en comités llamados **grupos de trabajo** (WG). Cada WG, pertenece a alguna área técnica del desarrollo de Internet2: Ingeniería, Middleware (interfaz software que provee funcionalidades rutinarias en una conexión típica Internet). Entre estas, como ejemplo se pueden mencionar las autenticaciones del usuario y aplicaciones. Cada una de estas áreas posee un director de área que es el responsable de las actividades de sus áreas respectivas. Los miembros de estos grupos de trabajo pueden ser tanto miembros de Internet2 como empresas de apoyo externo (por ejemplo, las empresas de apoyo económico).

Si un miembro de Internet2, tiene alguna idea a desarrollar entonces se debe contactar al director de área apropiada.
Los actuales grupos de trabajo por área, son:

- Ingeniería: IPv6, Measurement, Multicast, Network Management, Routing, Security, Topology.
- Middleware: MACE-Architecture, MACE-DIR (Directories), HEPKI-TAG (PKI Technical), HEPKI-PAG (PKI Policy).
- Applications: Arts and Humanities Initiative, Digital Imaging, Digital Video Initiative, Network Storage, Health Science Initiative, Research Channel, Video Conferencing (subcommittee Digital Video Initiative), Voice over IP.

Las dos primeras áreas, tienen labores que son transparentes al cliente / usuario, y que solo sirven para ofrecer un mejor servicio a las aplicaciones de la tercera área: (Aplicaciones). A partir de los nombres de los grupos de trabajo del área de aplicaciones, uno puede deducir a grandes rasgos de qué se trata. En el grupo de trabajo de red de almacenamiento; por ejemplo, se desarrolla la **infraestructura de almacenamiento distribuido en Internet2** (**I2-DSI**: Internet2 Distributed Storage Infrastructure). El objetivo de esto es el almacenar datos replicados a través de la red y cuando un cliente intente acceder a los datos, entonces, el sistema le provea los datos que se encuentran en el servidor más cercano (en la red) a él, manteniendo así, el tráfico lo más local posible.

a) RED INTERNET2

La red de Internet2 está compuesta por redes principales o troncales en USA, a los cuales se conectan los llamados gigaPoPs y troncales internacionales a los cuales a su vez se conectan gigaPoPs o nodos en particular como a las universidades. Un gigaPoP es una red regional (con ancho de banda del orden de los gigabits por segundo) conectada a la Internet2. Por ejemplo, en USA el MIT, la Universidad de Boston y la Universidad de Harvard conforman la capacidad **Gigabit en presencia de punto** (gigaPoP) llamado BOS.

A continuación, se muestra en la figura 1.3, esquema muy generalizado de la Internet2:

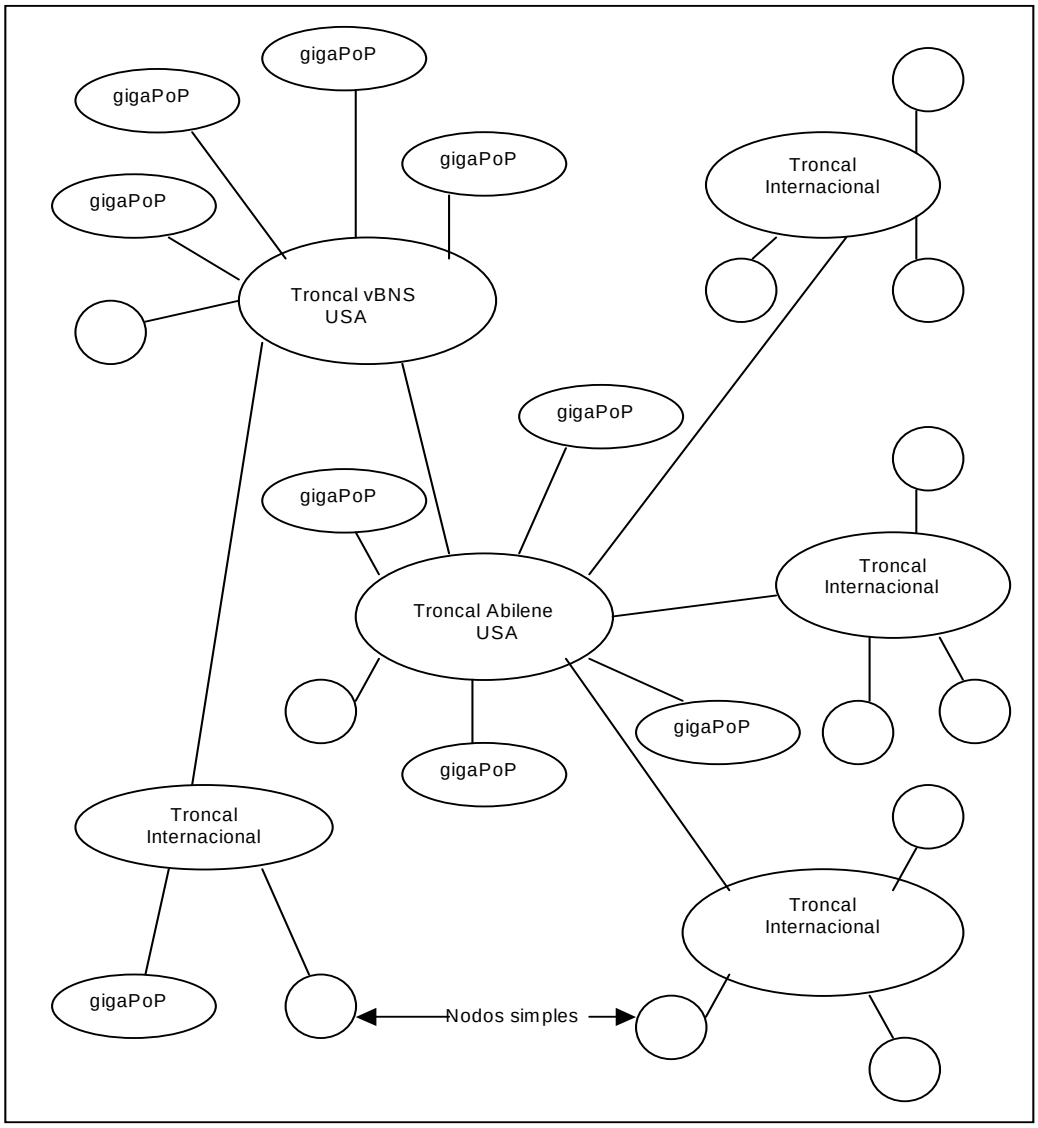


Figura 1.3. Esquema muy generalizado de la Internet2.

En la figura 1.3, se puede visualizar que actualmente existen dos grandes troncales en USA (aunque hoy en día el troncal Abilene es mucho mayor en ancho de banda, 2.4 Gbps o superior), de los cuales se distribuyen enlaces hacia troncales en otros países. Una de estas troncales internacionales, es la **Red Universitaria Nacional** (REUNA).

Para la conexión a la Internet2, es necesario algunos nuevos dispositivos o equipos y algunas nuevas conexiones por el lado de los clientes / usuarios de las respectivas universidades conectadas a la Internet2. Las troncales, son las responsables de encaminar el flujo de datos por Internet2 o Internet comercial según correspondan.

b) INTERNET2 DESPLAZARÁ A LA INTERNET COMERCIAL ACTUAL

El proyecto Internet2, reemplazará paulatinamente a la Internet comercial actual. Su objetivo es unir a las instituciones con recursos para desarrollar nuevas tecnologías y con mayores posibilidades; tal que posteriormente puedan extrapolarse a la Internet global. Las universidades mantendrán y continuarán teniendo un crecimiento sustancioso en el uso de las conexiones existentes de la Internet, que podrán seguir obteniendo de sus proveedores comerciales.

Aún más, el sector privado se beneficiará más con las aplicaciones y la tecnología desarrolladas por los miembros de Internet2. Hoy en día, las universidades e institutos de investigación han hecho inversiones y esfuerzos considerables encaminados a conectar la mayor parte de sus instalaciones a la Internet comercial; dicha inversión y esfuerzo no puede ser despreciado.

c) ARQUITECTURA DE INTERNET2

La red Internet2 consiste en múltiples infraestructuras interconectadas, cada uno con arquitecturas independientes. Las infraestructuras, son utilizadas por Internet2 para proporcionar servicios de red a los clientes / usuarios. La red Internet2, proporciona las siguientes amplias clases de servicios: servicio IP y servicio del circuito. El servicio IP, es una red IP que sustituye y engrandece la red de Abilene; y el servicio de circuito, proporciona circuitos punto a punto entre los puertos en la red Internet2. Estos circuitos proporcionan un ancho de banda garantizado y, gozan de un comportamiento determinado desde el punto de vista de los límites del Jitter, latencia y pérdida de paquetes. Se proporcionan tres diversos servicios del circuito: 1) Servicio estático del circuito. 2) Servicio dinámico del circuito. 3) Servicio de prueba HOPI. Hay cuatro infraestructuras arquitectónicas marcadas en la red Internet2, siguientes:

- Infraestructura base, consistiendo en los segmentos de fibra interconectados con el equipo de señales.
- Red IP.
- Infraestructura de conmutación multiservicio.
- Infraestructura HOPI testbed.

La figura 1.4, es un diagrama de la arquitectura total de la red Internet2. En el despliegue inicial, cada servicio será proveído por una infraestructura particular. Observar que este diagrama no muestra todos los nodos y conectores, sin embargo, muestra como los diferentes servicios interactúan arquitectónicamente. También,

demuestra algunas de las semejanzas y las diferencias entre los servicios. Se muestra en la 1.4 arquitectura de Internet2 siguiente:

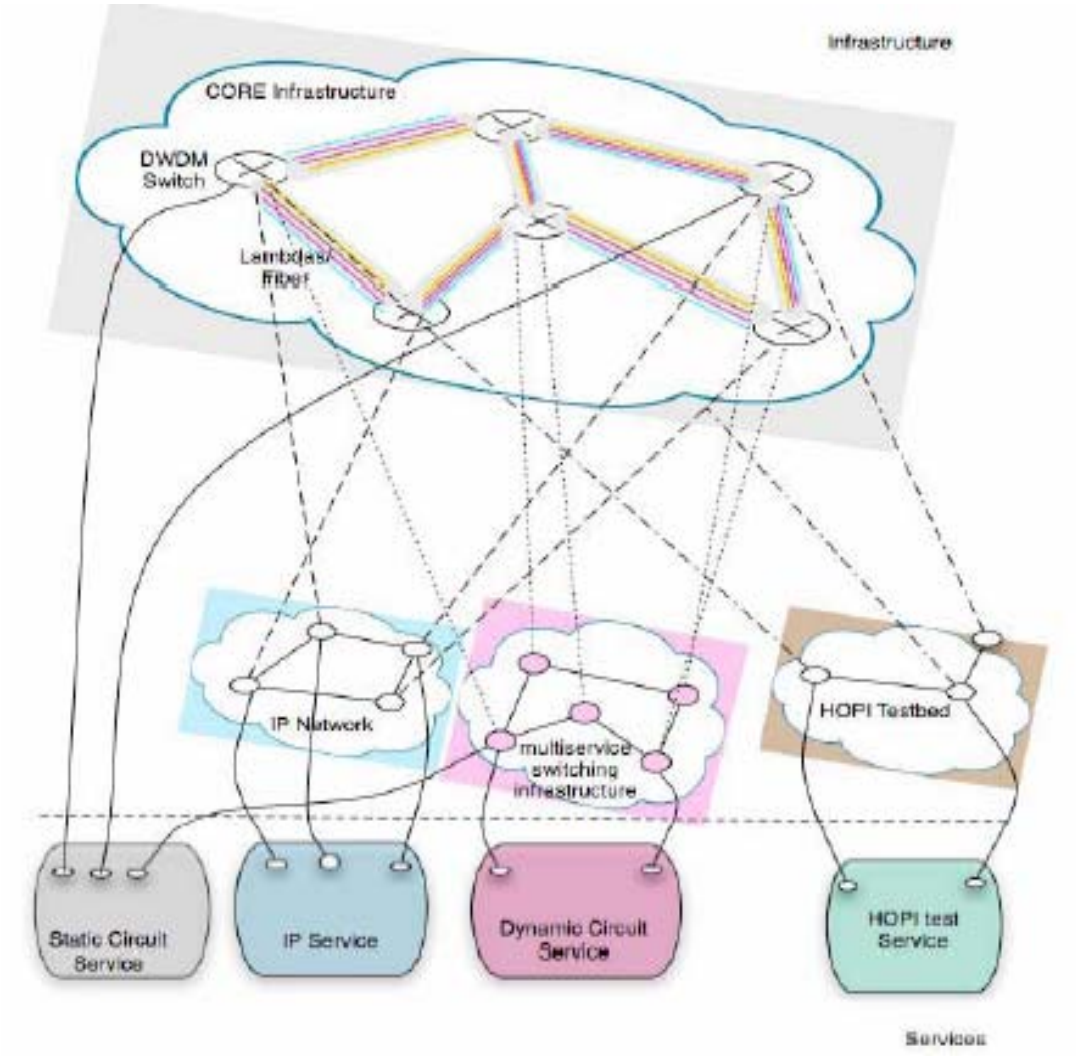


Figura 1.4. Arquitectura de Internet2.

d) INSTITUCIONES EDUCATIVAS QUE NO SON MIEMBROS DE LA INTERNET2

La participación en Internet2, está abierta para cualquier universidad que se comprometa a proveer facilidades para el desarrollo de aplicaciones avanzadas en su campus. La inversión financiera requerida para cumplir con estas obligaciones puede ser más de lo que muchas instituciones puedan permitirse por ahora. Sin embargo, la Internet2, tiene la intención de acelerar la transmisión de nuevas posibilidades a la comunidad mayor del sistema de redes. El costo de la tecnología utilizada y desarro-

llada por Internet2 descenderá a un nivel alcanzable a cualquier institución que actualmente tenga una conexión básica a Internet2 (Álvarez 2006).

1.3.2 LOCALIZADOR UNIVERSAL DE RECURSOS

El **Localizador Universal de Recursos** (URL), es una dirección de una página web; que permite ubicar como: archivo, una base de datos, una consulta u otro lugar en un servidor en cualquier lugar del mundo:

http://www.frontiertech.com/prodinfo.htm

El URL guía las consultas del visualizador, al servidor apropiado, utilizando distintos componentes de la dirección.

En el ejemplo anterior, la primera parte del localizador universal de recursos, es el protocolo de transferencia de hipertexto, indica que el visualizador envía una consulta utilizando el **Protocolo de Transferencia de Hipertexto** (HTTP), el protocolo de la web. HTTP permite realizar una consulta de red a un servidor web. El uso de la cadena inicial protocolo de transferencia de hipertexto: en lugar de **Protocolo de Transferencia de Archivos** (FTP) o Gopher, indica que el paquete de datos se envía a un servidor web. Sólo los servidores web serán capaces de interpretar el resto de la cadena de localizador universal de recursos. En el caso anterior, la computadora solicita al servidor que le permita acceder a un documento protocolo de transferencia de hipertexto. Los documentos de protocolo de transferencia de hipertexto se escriben en el **Lenguaje de Marcas Hipertexto** (HTML) o XML, que se describe con más detalle en la siguiente sección.

La segunda parte de una cadena del URL, es la doble barra ***//*****,** que indica que a continuación viene el nombre de una máquina.

La tercera parte de una cadena del URL, indica el tipo de máquina host. Cualquier servidor web remoto, suele tener el identificativo **telaraña alrededor del mundo** (WWW: World Wide Web). Como en el URL que se había indicado el uso de HTTP, lo que sigue es que la máquina es un servidor web y por tanto, al acceder a archivos remotos en la web de URL normalmente empieza con http://**www**.

Un URL se compone de tres partes principales:

1. Un identificador del servicio de **http**:
2. Un nombre de dominio del servidor.
   //www.frontiertech.com
3. Un nombre de ruta de acceso al servidor:
   /prodinfo.htm

Protocolo de Transferencia de Hipertexto (HTTP), es un método que emplea la WWW para mover información. Permite que el servidor remoto y la herramienta de navegación trabajen juntos. HHTP, indica que el visualizador envía una consulta utilizando el protocolo de transferencia de hipertexto, el protocolo en la web. HTTP, permite realizar una consulta de red a un servidor web. Es un protocolo, diseñado para responder a los requerimientos de los navegadores; para poder utilizarse, las

estaciones de trabajo han de estar configuradas TCP y Gopher (Álvarez 2006). Se muestran en las figuras 1.5 y 1.6 de HTTP siguientes:

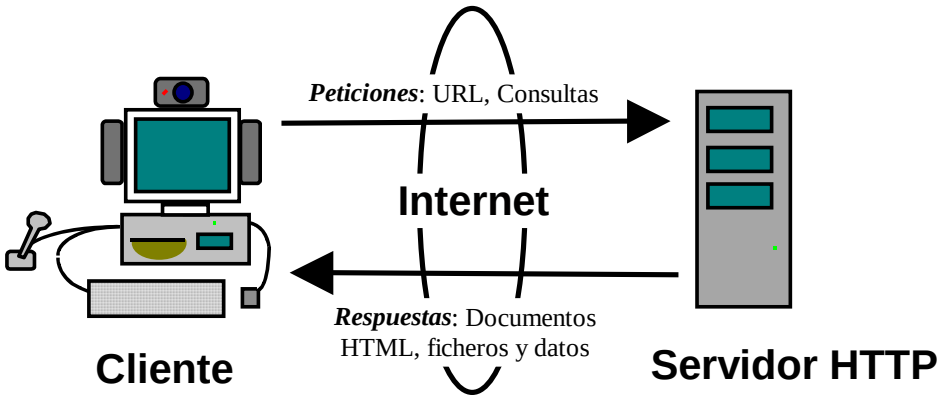


Figura 1.5. Comunicaciones en el servidor / cliente utilizando protocolo de transferencia de hipertexto.

Figura 1.6. Protocolo de Transferencia de Hipertexto.

1.3.3 VISUALIZADOR WEB

Un visualizador (o navegador) Web es el vehículo que permite a los clientes navegar en la *World Wide Web*. Los clientes / usuarios, tan sólo, escriben la dirección de una página web de un servidor específico del localizador universal de recursos, en el área de solicitud y, el visualizador Web, localiza el servidor web y solicita una página. El visualizador Web, espera normalmente unos segundos, hasta que se le envía de vuelta la información solicitada desde el servidor web. En ese momento el cliente puede ver la información en el visualizador.

Netscape Communicator y otras empresas, han desarrollado visualizadores y los han puesto en el mercado. Por ejemplo, el Internet Explorer de Microsoft; GNNWorks de GNN o Win tapestry de Frontier Technologies. Todos ellos, ofrecen un conjunto de funciones que ayudan al usuario en sus necesidades diarias e incluyen catálogos de direcciones para organizar las que se visitan más a menudo, correo electrónico, comercio electrónico, lectores de noticias y archivos de actualización de los proveedores de servicios de Internet. Se muestra en la figura 1.7. Primer Visualizador Mosaico (o navegador) siguiente:

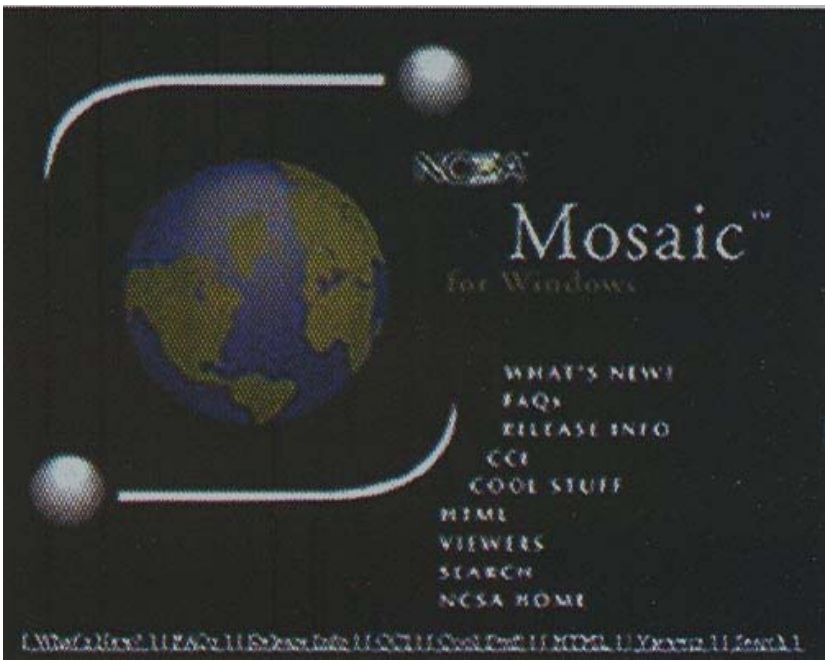


Figura 1.7. Primer visualizador o navegador.

La *World Wide Web* es una gigantesca serie de archivos, imágenes, sonidos o información almacenada en una enorme variedad de computadoras en todo el mundo. Para tener el acceso a la web y hacer posible que la computadora del usuario dis-

ponga de todas las capacidades necesarias, se requiere de un software especial llamado “visualizador web”.

1.3.4 DEFINICIÓN DE INTERNET

Una red es un conjunto de computadoras conectadas entre sí por conexión local o remota.

La **definición de Internet**. Es una red mundial de computadoras, es un conjunto de redes de computadoras conectadas entre sí por conexión local o remota. Y puertos de enlaces en el ámbito mundial que utilizan la colección de **Protocolo de Control de Transmisión / Protocolo de Internet** (TCP/IP) y protocolos inalámbricos, para comunicarse entre ellas. En el núcleo de Internet, se encuentra una red troncal de líneas de comunicación de datos de alta velocidad entre los nodos principales de servidores Web de Internet. Consistente en miles de sistemas informáticos comerciales, gubernamentales, educativos, investigaciones científicas y de otra naturaleza, que se encargan de dirigir los datos, audio, vídeo. La comunicación, se realiza a través de antenas, satélites artificiales, línea telefónica, fibra óptica, digital, inalámbrica, mediante un módem, por línea dedicada o conmutada. Hoy en día, se muestra en la figura 1.8 la Internet global siguiente:

Figura 1.8. Internet global.

1.3.5 WORLD WIDE WEB

El origen de la ***World Wide Web*** (WWW), significa en español “telaraña alrededor del mundo” de las páginas interactivas de Internet, se remonta al año 1989. En esa fecha el físico inglés, Tim Berners-Lee presentó un **sistema de comunicación**; basado en el uso de redes de computadoras, que permitía a científicos que estuvieran trabajando sobre una misma materia, separados normalmente por miles de kilómetros de distancia y acceder de inmediato a la información generada por sus colegas; así como a bases de datos, documentación, etcétera, sin necesidad de dar la vuelta al mundo (CIBERTEC 2000), se muestran en las figuras 1.9 la fotografía de Tim Berners-Lee y la figura 1.10 WWW siguientes:

En el año de 1993, el fenómeno de Internet, empezó a tener repercusión en los medios de comunicación, al tiempo que apareció el **primer visualizador gráfico de páginas *World Wide Web*.**

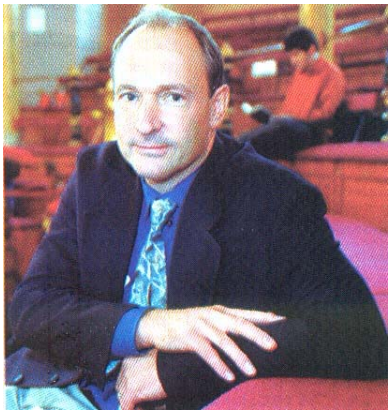


Figura 1.9. Fotografía de Tim Berners-Lee.

Figura 1.10. Telaraña alrededor del mundo.

La ***World Wide Web*** (WWW) es un conjunto de documentos ligados entre sí y ubicados en los servidores web de Internet. Gracias a la vinculación entre sí de documentos afines, la web facilita enormemente la localización de información por parte de los clientes / usuarios. La web posee bases de datos para almacenar información. Cada uno de ellos, posee un nombre especial, llamado **Localizador Universal de Recursos** (URL), o simplemente una dirección web.

Las redes interconectadas no necesitan estar en la misma ubicación geográfica ni en el mismo edificio; pueden estar en lugares remotos entre sí, desde el punto de vista físico, con conexiones que utilizan líneas de datos de propósitos especiales, como puede ser radio, por satélite, enlaces de radio infrarrojo, TV por cable o incluso líneas telefónicas ordinarias o módems, fibras ópticas. Las computadoras remotas parecen convertirse en locales y permiten transferencias de: archivos, correo electrónico, comercio electrónico, etcétera; uso compartido de impresoras, discos magnéticos y otras características que incluyen el acceso a *World Wide Web*.

En el paradigma de la Web, la máquina cliente accede y visualiza páginas web de servidores mediante un visualizador. Desde la perspectiva del visualizador del cliente / usuario, un hiperenlace es una palabra resaltada, que cuando se selecciona con el ratón transfiere al usuario a otra página web en la misma máquina o en cualquier otra en Internet. Esta capacidad para enlazar páginas de información entre diferentes sistemas de computadoras de la web es una de sus más importantes funciones. Tras la página web, el **Lenguaje de Marcas Hipertexto** (HTML), define una dirección de una computadora host de la web a la que se enlaza mediante la palabra resaltada. Seleccionando el hiperenlace se accede a la computadora y, el archivo especificado en dicha dirección y se descarga la información en la computadora del cliente.

Ahora, es muy común que centros educativos, noticias, noticieros de televisión, eventos deportivos y hasta películas incluyan una dirección web, etcétera; como http://www.unfv.edu.pe, se muestra en la figura 1.11 con dirección en la web siguiente:

http://www.unfv.edu.pe

Figura 1.11. Con dirección en la web.

1.3.6 SERVIDOR WEB

Es el servidor de Internet al que se conectan otros servidores como: servidor intranet, servidor extranet, servidor Proxy y otros servidores, es la computadora principal, que permite compartir los recursos del sistema durante procesamiento de datos. Por ejemplo, cuando se comunican dos programas por la red, el cliente es el que inicia la comunicación y el programa que espera ser contactado es el servidor. Cada programa puede actuar como servidor para un servicio y como cliente para otro.

Es un sistema de la computadora, que controla el acceso a una red y a los recursos de la red como las impresoras y el compartir archivos. Algunos servidores proporcionan acceso a la información almacenada en bases de datos o puesta en sitios de la red mientras que otros coordinan el flujo de datos, de los procesos entre otros servidores y los sistemas de respaldo. Un objeto que proporciona servicios que se utilizan por otros objetos. Los objetos que utilizan los servicios son clientes.

La mayoría de los protocolos de Internet, se comunican bajo el principio de cliente / servidor. Por ejemplo, el servidor de noticias, proporciona información al cliente. El

cliente de un lector de noticias, proporciona información al servidor. La información es cómo un modelo en la Web que proporciona información confidencial o pública a los clientes / usuarios.

Los servidores Web se utilizan principalmente para mantener un directorio de páginas y lugares web y, responder a las consultas de los visualizadores web para ver esas páginas e interactuar con el servidor.

Por ejemplo, un servidor Web es un servidor de archivos UNIX, una computadora central, host, Linux, Windows Vista o un servidor Windows 2003 configurado con el hardware y software apropiado para responder a las consultas de los clientes o usuarios, mediante un visualizador. Esencialmente, un servidor Web descarga páginas y aplicaciones web hacia los usuarios.

Si, el servidor Web deja de dar servicio a los clientes / usuarios, éstos serán incapaces de acceder a las páginas web de dicho lugar hasta que vuelva a estar disponible.

Antes de que un visualizador de un cliente, pueda descargar y visualizar páginas del lugar seleccionado el servidor web está en espera de consultas de los visualizadores web. Cuando recibe una consulta de descarga o una “vista” de un visualizador, busca el documento o el lugar pedido y lo envía de vuelta al visualizador para que el cliente pueda revisarlo. En ese momento, la función del servidor web es responder al visualizador obteniendo las páginas o lugares solicitados.

La clave de un sistema intranet, basada en Web es el servidor web. Con un servidor web de una empresa / institución en particular, se puede publicar información pública o confidencial utilizando documentos visualmente atractivo, ricos en contenidos, formados por texto, gráficos, vídeo y sonido. Los lectores pueden acceder a la información desde sus computadoras usando un visualizador web, que se “sirven” bajo demanda cuando un cliente solicita información. El servidor web también puede ejecutar aplicaciones de respaldo que hacen de interfaz con bases de datos y otras aplicaciones. Por ejemplo, realizar auto notificación de correo, la información del servidor web se puede administrar y distribuir mediante herramientas de administración.

Algunos programas de servidores Web permiten configurar niveles de seguridad multicapa y administrarlos desde una computadora personal. En otras palabras, se pueden fijar distintos niveles de acceso a diferentes oficinas o departamentos, personas dentro de la empresa y fuera de ella. Incluso, es posible fijar los permisos de lectura y escritura desde el administrador de archivos de programa, de forma, que diferentes carpetas y archivos tengan distintos niveles de acceso. La posibilidad de fijar dichos niveles permite que cierta información pueda llegar al público general, a la vez que se protege la información confidencial. En la configuración de una LAN típica, también, se dispone de esta posibilidad. Se muestra en la figura 1.12 servidores web, siguiente:

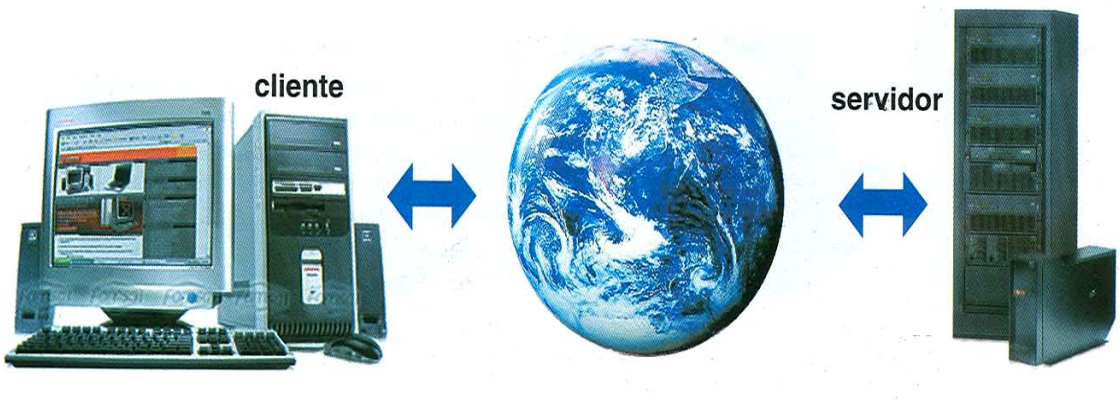


Figura 1.12. Servidores web.

1.3.7 DEFINICIÓN DE UN SISTEMA INTRANET

Desde principios de la década de 1980, las organizaciones con redes privadas han emprendido la tarea de conectar diferentes tipos o marcas de computadoras, como PC, IBM, Macintosh y máquinas basadas en UNIX, para compartir información. Antes, los administradores de redes tenían que lidiar con la incompatibilidad de hardware y software. Para eliminar estas barreras en la comunicación electrónica, las empresas tuvieron que estandarizar plataformas de hardware y software específicos para garantizar una conectividad eficiente.

Definición de una intranet. Es una red privada de servidor web / servidor intranet, es una red, diseñada para el procesamiento de información dentro de una empresa / institución. Las intranets, utilizan las tecnologías de Internet y las herramientas de la web, para publicar información y proporcionar a los clientes / usuarios de la organización acceso a esta información. Es una red privada de conexión entre computadoras que utilizan los protocolos estándar de Internet, sin embargo, están protegidos por un identificador de cliente / usuario. Es una red concebida para organizar y compartir la información, así como, para efectuar transacciones digitales dentro de una empresa / institución. Técnicamente, es un sitio privado al que sé accede por claves y utiliza aplicaciones asociadas a Internet como páginas web, exploradores, correo electrónico, comercio electrónico, grupos temáticos y listas de correos, sin embargo, todo ello es accesible únicamente a quienes forman parte de la organización (Álvarez 2006). Se muestra una intranet en la figura 1.13, donde se observa cómo un servidor intranet puede atender a varios clientes / usuarios de un hospital.

Intranet es un sistema de administración empresarial o corporativa computarizado, basado en el protocolo TCP/IP o WAP, para la comunicación entre las máquinas que participan en la red para tal propósito.

Por ejemplo, para nuestra empresa / institución se requiere los servicios principales que debe proporcionar un sistema intranet que son: sistemas de gestión; administración, navegación, compartir información, administrar información, mensajería, correo electrónico, etcétera.

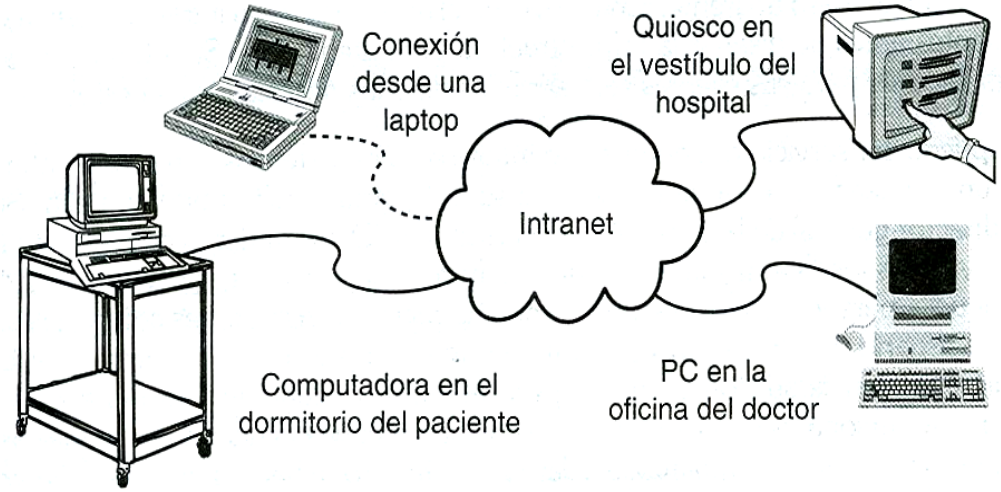


Figura 1.13. Servidor intranet que atiende a varios clientes / usuarios.

1.3.8 COMPONENTES DE HARDWARE Y SOFTWARE PARA INTRANET E INTERNET

Las intranets están basadas en la tecnología de Internet, construida principalmente alrededor del protocolo TCP/IP. Actualmente, los **administradores de sistemas de información** (MIS), deberán elegir entre las tecnologías competitivas que hacen uso de Internet y una posible intranet en la institución o empresa, se hace un breve resumen de los componentes más importantes de Internet e intranet, que se muestra en la tabla 1.1 siguiente:

Navegadores web	Netscape Communicator Corp.
	Internet Explorer de Microsoft, etc.
Máquina cliente para web	UNIX, PC, Mac, HP, Dell, etc.
Software para servidor intranet	Netscape Enterprise, Lotus, IIS de Micro-soft, etc.
Sistemas operativos	UNIX, Windows 2003, NetWare, etc.
Para servidor web	UNIX, Windows 2003, NetWare, Mac,
	Windows vista, etc.
Protocolos de red	TCP/IP, IPX/SPX, WAP, etc.
Sistemas operativos para red	UNIX, Windows, NetWare, OS/2, etc.
Hardware para servidores	IBM, PC, Mac, HP, AS/400, etc.
Tecnología de redes	Ethernet, Token Ring de IBM, etc.
Topologías físicas	Topología estrella, bus, anillo, etc.

Tabla 1.1. Componentes más importantes de Intranet e Internet.

CAPÍTULO II

2. ALGORITMOS GENÉTICOS

2.1 INTRODUCCIÓN

Los Algoritmos Genéticos (AGs), fueron creados por John Holland (1975). Es una técnica de búsqueda aleatoria dirigida, que puede encontrar la solución óptima global en los espacios de búsqueda multidimensionales complejos. Los Algoritmos genéticos, es un modelo de la evolución natural que los operadores emplean y que está inspirado por el proceso de evolución natural. Estos operadores conocidos como los operadores genéticos, manipulan a los individuos en una población a lo largo de varias generaciones para mejorar su aptitud gradualmente. Como se tratará en la próxima sección, los individuos son como la población semejante a los cromosomas y es normalmente representado como una cadena de números binarios.

La evolución de una población de individuos, se gobierna por el "teorema del esquema" de la población, por lo que se refiere a la similitud de los bits con ciertas posiciones de esos individuos. Por ejemplo, el esquema 1*0* que describe un conjunto de individuos, cuyo primer y tercer bits son 1 y 0, respectivamente. Aquí, * representa los medios que cualquier valor sería aceptable. En otros términos, los valores de bits en las posiciones marcadas con *, podría ser: 0 o 1 en una cadena binaria. Un esquema se caracteriza por dos parámetros: la longitud definida y el orden. La longitud definida, es la longitud entre los primeros y últimos momentos con los valores fijos. El orden de un esquema, es el número de bits con los valores especificados. Según el teorema del esquema, la distribución de un esquema a través de la población de una generación a la próxima depende de su orden, mientras se ha definido longitud y aptitud.

Los Algoritmos genéticos no utilizan mucho conocimiento sobre el problema que va a perfeccionar y no trata directamente con los parámetros del problema. Trabajan con códigos que representan los parámetros.

Así, el primer problema en una aplicación de algoritmos genéticos, es cómo codificar el problema bajo estudio, es decir, cómo representar los parámetros del problema.

Los algoritmos genéticos operan con una población de posibles soluciones, no sólo es una posible solución, y el segundo problema es por consiguiente, cómo crear la población inicial de posibles soluciones.

El tercer problema en una aplicación de algoritmos genéticos, es cómo seleccionar o crear un conjunto adecuado de operadores genéticos.

Finalmente, como ocurre con otros algoritmos de búsqueda, los algoritmos genéticos, tienen que conocer la calidad de soluciones encontradas para mejorarlas más adelante. Hay una necesidad, por consiguiente, de una interfaz entre el ambiente del

problema y los propios algoritmos genéticos y también, existe una necesidad de poseer conocimiento de algoritmos genéticos. El diseño de esta interfaz puede considerarse como el cuarto problema.

2.2 EVOLUCIÓN BIOLÓGICA

Todo organismo vivo consiste de células. En cada célula hay el mismo conjunto de cromosomas. Los cromosomas son cadenas de ADN y sirven como modelo del organismo completo. Un cromosoma consiste de genes, bloques de ADN. Cada gen, codifica una proteína particular.

Básicamente, podría decirse, que cada gen codifica una característica, Por ejemplo, el color de los ojos de una persona, los posibles valores de una característica (azul, cafés) se llama alelos. Cada gen tiene una propia posición en el cromosoma, esta posición se denomina locus.

El conjunto completo del material genético (todos los cromosomas) se llama genoma. Un conjunto particular de genes en el genoma es llamado genotipo. El genotipo va acorde con el desarrollo posterior, es como la base del nacimiento del fenotipo del organismo, el cual es la característica (física o mental), como el color de los ojos de la persona, la inteligencia de la persona, etcétera (Darwin 1930).

Durante la reproducción, lo primero que ocurre es la **recombinación / cruzamiento** (crossover). Los genes padres, se combinan para formar un nuevo cromosoma. El retoño generado, puede mutar. Las mutaciones son pequeñas variaciones en los elementos del ADN. La aptitud de un organismo que se mide por el éxito de dicho organismo a lo largo de su vida para sobrevivir. http://www.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/AlgoritmosGeneticos.pdf
Se muestra en la figura 2.1 célula y la tabla 2.1 significado de valores siguientes:

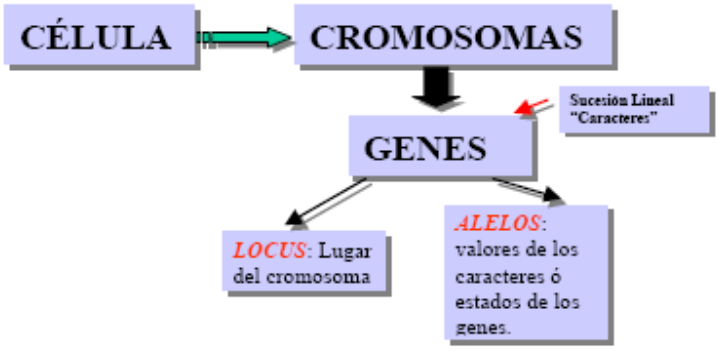


Figura 2.1. Célula.

Algoritmo Genético	Significado
Cromosomas (cadena, individuo)	Solución (código)
Genes (bits)	Parte de la solución.
Locus	Posición del Gen
Alelos	Valor del Gen
Fenotipo	Solución Decodificada (Apariencia Externa)
Genotipo	Solución Codificada *(Estructura Interna)*

Tabla 2.1. Significado de valores.

A lo largo de las generaciones, las poblaciones, evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados de Charles R. Darwin. Se muestra en la figura 2.2 fotografía de Charles Robert Darwin.

En la naturaleza, los individuos de una población compiten entre sí, en la búsqueda de recursos tales como: comida, agua y refugio. Incluso los miembros de una misma especie compiten por el número de descendientes. Por el contrario, individuos poco dotados producirán un menor número de descendientes. Esto significa, que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes antepasados, puede a veces producir descendientes “súperindividuos”, cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

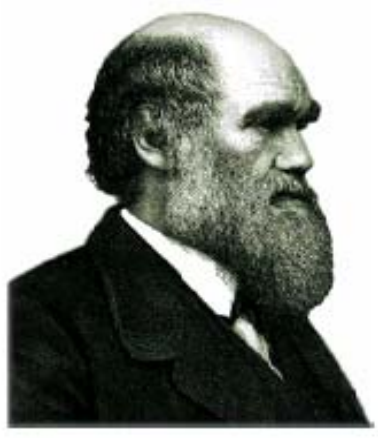


Figura 2.2. Fotografía de Charles Robert Darwin.

2.3 REPRESENTACIÓN

Normalmente, se representa la optimización de los parámetros que van a ser perfeccionados en un formulario de una cadena de operadores genéticos que son adecuados para este tipo de representación. El método de representación tiene un gran impacto en la actuación de algoritmos genéticos. Los esquemas de la representación diferentes podrían causar las actuaciones diferentes lo que se refiere a la exactitud y tiempo del cómputo.

Hay dos métodos de representación más comunes para los problemas de optimización numérica (Davis 1991).

El método preferido, es el método de representación de cadena binaria. La razón para que este método sea popular es que el alfabeto binario ofrece el número máximo esquemático por binarios en comparación con otras técnicas codificadas. Varios binarios que se codifican bajo los esquemas de representación, por ejemplo, el código uniforme y el código de escala de Gray.

El segundo método de representación, es utilizando un vector de enteros o números reales, con cada entero o número real representado sólo el parámetro. Cuando un esquema de la representación binaria es empleado, un problema importante es decidir el número de bits utilizados en el código de parámetros que se va a optimizar. Cada parámetro, debería codificarse con el número óptimo de bits que cubren todas las posibles soluciones, cuando se utilizan pocos o demasiados bits que podría afectar el rendimiento del sistema.

2.4 CREACIÓN DE LA POBLACIÓN INICIAL

Al inicio de la optimización, los algoritmos genéticos, requieren un grupo de soluciones iniciales o población inicial. Hay dos maneras, de formar esta población inicial. El primero consiste en utilizar las soluciones aleatorias producidas al crear un generador del número aleatorio. Este método es preferido para los problemas que no tiene a priori el conocimiento que existe o por evaluar la actuación de un algoritmo.

El segundo método, emplea el conocimiento a priori sobre el problema de optimización dado. Utilizando este conocimiento, se obtiene un conjunto de requerimientos y se coleccionan soluciones que satisfacen esos requerimientos para formar una población inicial. En este caso, los algoritmos genéticos, empiezan la optimización con un conjunto de soluciones aproximadamente conocidas y, por consiguiente, converge a una solución óptima en menos tiempo que con el método anterior.

2.5 OPERADORES GENÉTICOS

El diagrama de flujo de algoritmos genéticos simple, se muestra en la figura 2.3. Hay tres operadores genéticos comunes: de **selección**, de **cruzamiento**, y de **mutación**. El operador de reproducción adicional es el de la inversión. También, a veces,

se aplican algunos de estos operadores que se basan en la evolución natural, y en la literatura sobre Algoritmos genéticos pueden encontrarse muchas versiones de estos operadores. No es necesario emplear todos estos operadores en un algoritmo para realizar la optimización de un problema, porque cada una de las funciones es independiente de las otras. La opción o plan de operadores dependen del problema y la representación formal de los planes empleados. Por ejemplo, no puede utilizarse operadores diseñados para las cadenas binarias directamente en cadenas codificadas con enteros o números reales.

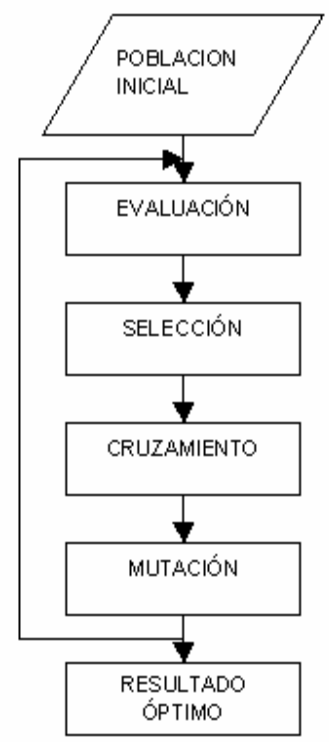


Figura 2.3 Diagrama de flujo de algoritmos genéticos simple.

a) LA SELECCIÓN

El objetivo del procedimiento de la selección, es producir más reproducciones de individuos cuyos valores de aptitud sean más altos que aquellos cuyos valores de aptitud son bajos. El procedimiento de la selección, tiene una influencia significante en manejar la búsqueda hacia un área prometedora y las soluciones buenas encontradas en un corto tiempo. Sin embargo, debe mantenerse la diversidad de la población y evitar la convergencia prematura y se debe alcanzar la solución óptima global. En los algoritmos genéticos, hay dos procedimientos de selección importantes: la selección proporcional y la selección de clasificación jerárquica, basada en (Whitley 1989).

La selección proporcional, normalmente se llama la "**rueda de la ruleta**", la selección por este procedimiento, es un mecanismo que recuerda el funcionamiento de una rueda de la ruleta. Mediante este procedimiento se valora la aptitud de individuos los que son representados por las anchuras de las hendiduras de la rueda. Para seleccionar a un individuo para la próxima generación, los individuos de las hendiduras con anchuras grandes representan los valores de aptitud altos, estos tendrán una oportunidad más alta de ser seleccionados, después, de un funcionamiento de la rueda se obtiene un conjunto aleatorio.

Una manera, de prevenir la convergencia rápida es limitar el rango de ensayos asignados a un solo individuo, para que ningún individuo genere demasiados descendientes. La clasificación jerárquica, tiene como base el procedimiento de la producción de idea planteada. Según este procedimiento, cada individuo genera un número esperado de descendientes que se basa en la línea de su valor de aptitud.

b) CRUZAMIENTO

Se considera que esta operación es lo que hace a los algoritmos genéticos diferente de otros algoritmos, como la programación dinámica. Esta operación de cruzamiento, se utiliza para crear dos nuevos individuos (niños) de dos individuos existentes (padres), que son escogidos de la población actual para la operación de selección. Hay varias maneras de hacer esta operación. Algunas operaciones de cruzamientos más comunes son: cruzamiento basado en un punto, cruzamiento basado en dos puntos, cruzamiento de ciclo y cruzamiento uniforme.

El cruzamiento basado en un punto. Es la operación de cruzamiento más simple. Se seleccionan dos individuos al azar como los padres de una familia de individuos formados por el procedimiento de selección, el procedimiento se corta al azar en un punto escogido. Las colas son las partes después del punto cortante, y dos nuevos individuos (niños) se producen. Se observa que esta operación no cambia los valores de los bits. Se muestran los ejemplos en la tabla 2.2 especificados de (i, ii, iii, iv) para las operaciones de cruzamientos diferentes.

TABLA 2.2. Cruzamiento de cromosomas.
i) Un cruzamiento basado en un punto.

Padre 1	10001001111
Padre 2	01101100011

La cadena nueva 1	10001100011
La cadena nueva 2	01101001111

De (i) El cruzamiento basado en un punto, el padre 1, tiene once bits de ceros y unos. Sea la longitud de 11 bits, como la fórmula L-1 = L puntos de cruce, y aplicando en (i) se tiene 11-1 = 10 puntos; entonces, se debe generar un número aleatorio de 1 al 10. Está vez, se ha tomado el número 5, es decir, del padre 1 se toma los 5 bits primeros que son: 10001 y del padre 2, se toma los 6 bits últimos que son: 100011 y se une ambos bits; entonces se tiene la cadena nueva 1 que son los siguientes bits: 10001100011 resultado del cruzamiento basado en un punto. Del mismo modo, del padre 2 se toma los 5 primeros bits que son: 01101 y del padre 1 se toma los 6 últimos bits que son: 001111 y se une ambos bits; entonces, se tiene la cadena nueva 2 que son el siguiente conjunto de bits: 01101001111 resultado del cruzamiento basado en un punto. De está manera, en general se realizan el cruzamiento de puntos de la población de los algoritmos genéticos.

ii) Cruzamiento basado en dos puntos.

Padre 1	10100011010
Padre 2	01101111011

La cadena nueva 1	10101111010
La cadena nueva 2	01100011011

iii) El cruzamiento de ciclo.

Padre 1	12345678
Padre 2	abcdefgh

La cadena neuva 1	1b3de6g8
La cadena neuva 2	a2c45f7h

iv) El cruzamiento uniforme.

Padre 1	1001011
Padre 2	0101101
Plantilla	1101001

La cadena nueva 1	1001101
La cadena nueva 2	0101011

c) LA MUTACIÓN

En este procedimiento, se verifican a todos los individuos de la población, bit por bit, y los valores de bits se invierten al azar según una proporción especificada. A diferencia de la operación de cruzamiento, la mutación es una operación monádica, es decir, una cadena de un niño, se produce de una sola cadena de un padre. El operador de la mutación obliga al algoritmo a investigar nuevas áreas de solución. En el futuro, ayuda a los algoritmos genéticos a evitar la convergencia prematura y encuentra la solución óptima global. Un ejemplo se muestra en la tabla 2.3 siguiente:

TABLA 2.3 Mutación de cromosomas.

La cadena vieja	11000101110
La cadena nuena	11001101110

Se puede tener uno o más operadores de mutación, para realizar la representación; introducir la variable genética en la población a través de la mutación. Se muestra en la figura 2.4 mutación del individuo.

Algunos aspectos importantes a tener en cuenta son:

- Debe permitir alcanzar cualquier parte del espacio de búsqueda.
- El tamaño de la mutación debe ser controlado.
- Debe producir cromosomas válidos.

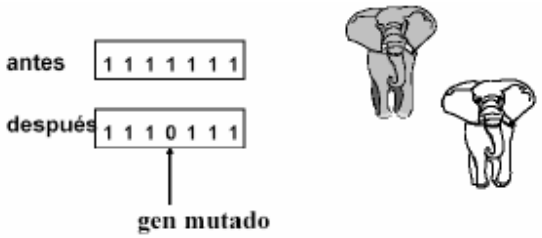


Figura 2.4. Mutación del individuo.

La inversión, este operador adicional es empleado para varios problemas descritos en este tema, incluso en el problema de la colocación celular, problemas de diseño y el problema del viajante de comercio. También, opera a un individuo en un momento dado. Se selecciona dos puntos al azar de un individuo y la parte de la cadena entre esos dos puntos se invierte, según se muestra en la tabla 2.4 siguiente:

La cadena vieja	1011001110
La cadena nuena	1000111110

Tabla 2.4 Inversión de un segmento de la cadena binaria.

2.6 PARÁMETROS DE CONTROL

Los parámetros de control importantes de los algoritmos genéticos simples, incluyen el tamaño de la población y el número de individuos en la población; los cruzamientos tasan y proporcionan la mutación. Varios investigadores han estudiado el efecto de estos parámetros de rendimiento de los algoritmos genéticos (Schaffer 1989), (Grefenstette 1986, Fogarty 1989). Las conclusiones principales, han sido considerando como parámetro de control los tamaños de población: grandes para el manejo simultáneo de varias soluciones e incremento del tiempo de cómputo por la iteración; sin embargo, desde muchas muestras del espacio de búsqueda se ha visto que la probabilidad de convergencia a una solución óptima global es más alta que al usar un tamaño pequeño de la población. La proporción de cruzamiento, determina la frecuencia de operación de cruzamiento. Es útil, en la salida de optimización al descubrir una región prometedora. Una frecuencia de cruzamiento baja disminuye la velocidad de convergencia a un área de solución. Si la frecuencia es demasiado alta, lleva a la saturación alrededor de una solución. La operación de mutación se controla por la proporción de mutación. Una proporción de mutación es alta si introduce la diversidad alta en la población y podría causar la inestabilidad. Por otro lado, normalmente, es muy difícil para los algoritmos genéticos encontrar una solución óptima global con una proporción demasiado baja de mutación.

2.7 FUNCIÓN DE EVALUACIÓN DE APTITUD

La unidad de evaluación de aptitud actúa como una interfaz entre los algoritmos genéticos y el problema de optimización. Los algoritmos genéticos, evalúan las soluciones para mejorar su calidad según la información producida por esta unidad y no utilizando la información directa sobre la estructura. Se diseña el plan de problemas, los requisitos funcionales que son especificados por el diseñador, este plan tiene que producir una estructura de rendimiento que realiza las funciones deseadas dentro de las restricciones predeterminadas. La calidad de una solución, propuesta es usualmente calculada dependiendo del resultado de la solución que realiza las funciones deseadas y satisface las restricciones dadas. En el caso de los algoritmos genéticos, este cálculo, debe ser automático y el problema es cómo inventar un procedimiento que calcule la calidad de soluciones.

Las funciones de evaluación de aptitud, podrían realizarse manualmente dependiéndolo de cuan complejos o cuan simple sea el problema de optimización. Donde una ecuación matemática no puede formularse para esta tarea, se puede construir un procedimiento basado en reglas, para usarla como una función de aptitud o en algunos casos en que los dos pueden combinarse. Donde hay pocas restricciones

que son muy importantes y no pueden violarse, las estructuras o soluciones pueden eliminarse de antemano diseñando apropiadamente el esquema de la representación.

2.8 FUNCIONAMIENTO DE ALGORITMOS GENETICOS

Se parte de una función f(x) muy sencilla de:

$$f(x) = x^2$$

Sí se desea encontrar el valor de x, que hace que la función f(x) alcance su valor máximo, sin embargo, restringiendo a la variable x, a tomar valores comprendidos entre 0 y 31. Aún más, a x sólo, le vamos a permitir tomar valores enteros, es decir: 0, 1, 2, 3,..., 30, 31. Obviamente el máximo se tiene para x = 31, donde f, vale 961. No se necesita tener conocimientos de algoritmos genéticos para resolver este problema, sin embargo, su sencillez hace que el algoritmo sea más fácil de comprender (Muñoz 2005).

Lo primero, que se debe hacer es encontrar una manera de codificar las posibles soluciones (posible valores de x). Una manera de hacerlo es mediante la codificación binaria. Aplicando esta codificación a un posible valor de x es:

(0, 1, 0, 1, 1)

¿Cómo se interpreta esto? Muy sencillo: multiplicar la última componente (un 1) por 1, la penúltima (un 1) por 2, la anterior (un 0) por 4, la segunda (un 1) por 8 y la primera (un 0) por 16 y a continuación calcular la suma: Resulta 11. Observar que (0, 0, 0, 0, 0) equivale a x = 0 y que (1, 1, 1, 1, 1) equivale a x = 31. Por ejemplo, se muestra en la figura 2.5, la operación de cromosoma siguiente:

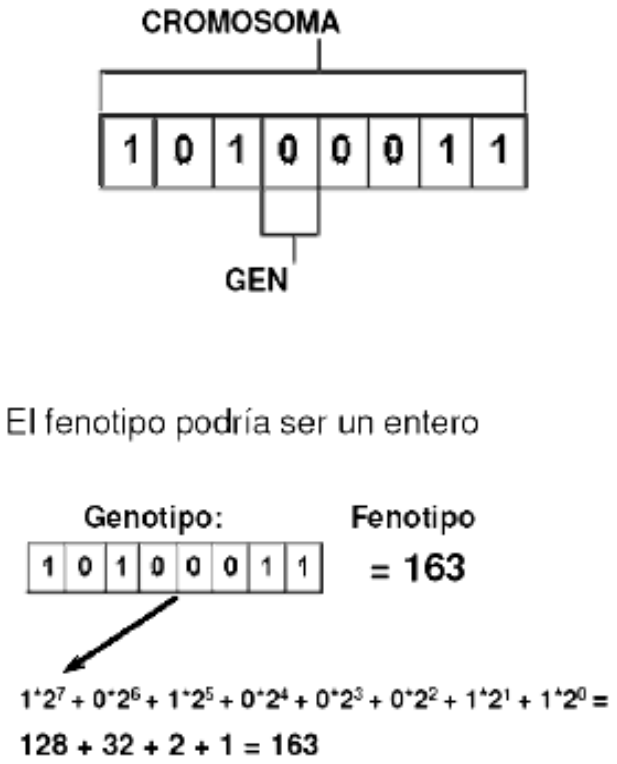


Figura 2.5. Operación de cromosoma.

A cada posible valor de la variable x, en representación binaria se le llamará individuo. Una colección de individuos constituye lo que se denomina población y el número de individuos que la componen es el tamaño de la población.

Una vez que se tiene codificada la solución, se debe escoger un tamaño de población. Para este ejemplo ilustrativo, se escogerá a 6 individuos.

Se debe partir de una población inicial. Una manera de generarla es aletoriamente: Se coge una moneda y se lanza al aire; si sale cara, la primera componente del primer individuo es un cero (0) y en caso contrario si sale una cruz es un uno (1). Repetir el lanzamiento de la moneda y se tendrá la segunda componente del primer individuo (un 0 si sale cara y un 1 si sale cruz). Así hasta 5 veces y se obtendrá el primer individuo. Repetir ahora la secuencia anterior para generar los individuos de la población restante. En total se tiene que lanzar 5 * 6 = 30 veces la moneda.

El siguiente paso es hacer competir a los individuos entre sí. Este proceso se conoce como selección. La tabla 2.5, resume el proceso.

TABLA 2.5. SELECCIÓN				
(1)	(2)	(3)	(4)	(5)
1	(0, 1, 1, 0, 0)	12	144	6
2	(1, 0, 0, 1, 0)	18	324	3
3	(0, 1, 1, 1, 1)	15	225	2
4	(1, 1, 0, 0, 0)	24	576	5
5	(1, 1, 0,1, 0)	26	676	4
6	(0, 0, 0, 0, 1)	1	1	1

Cada fila en la tabla 2.5, está asociada a un individuo de la población inicial. El significado de cada columna de la tabla es el siguiente:

(1) = Número que se le asigna al individuo.
(2) = Individuo en codificación binaria.
(3) = Valor de x.
(4) = Valor de f(x).

Observar que el mejor individuo es el de la fila 5 con f = 676. Calculando la media de f, se obtendrá fmed =324.3.

En cuanto a la columna (5), se realiza la comparación entre parejas de los individuos. Una manera de realizar el proceso de selección es mediante un torneo entre dos. A cada individuo de la población se le asigna una pareja y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La columna (5), indica la pareja asignada a cada individuo, lo cual se ha realizado aletoriamente. Existen variantes de este proceso de selección, aunque este método es válido para ilustrar el ejemplo.

Después, de realizar el proceso de selección, la población que se tiene es la mostrada en la columna (2), de la tabla 2.6. Observar, por ejemplo, que en el torneo entre el individuo 1 y el 6 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo cae en el olvido.

TABLA 2.6. CRUCE			
(1)	(2)	(3)	(4)
1	(0, 1, 1, 0, 0)	5	1
2	(0, 1, 1, 0, 0)	3	3
3	(1, 0, 0, 1, 0)	2	3
4	(1, 0, 0, 1, 0)	6	1
5	(1, 1, 0, 1, 0)	1	1
6	(1, 1, 0, 1, 0)	4	1

Tras realizar la selección, se realiza el cruce. Una manera de hacerlo es mediante el cruce de individuos: se forman parejas entre los individuos aleatoriamente de forma similar a la selección. Dados dos individuos pareja se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos 1). Por ejemplo, en la pareja 2-3 el punto de cruce es 3, lo que significa que un hijo de la pareja conserva los tres primeros bits del padre y hereda los dos últimos de la madre, mientras que el otro hijo de la pareja conserva los tres primeros bits de la madre y hereda los dos últimos del padre. La población resultante se muestra en la columna (2), de la tabla 2.7 (Muñoz 2005).

TABLA 2.7. POBLACIÓN TRAS CRUCE			
(1)	(2)	(3)	(4)
1	(0, 1, 0, 1, 0)	10	100
2	(1, 1, 1, 0, 0)	28	784
3	(0, 1, 1, 1, 0)	14	196
4	(1, 0, 0, 0, 0)	16	256
5	(1, 1, 0,1, 0)	26	676
6	(1, 0, 0, 1, 0)	18	324

En la columna (3), se tiene el valor de x; en la siguiente se tiene el valor de f correspondiente.

Veamos que ahora, el valor máximo de f es 784 (para el individuo 2), mientras que antes, de la selección y el cruce era de 676. Además, fmed ha subido de 324.3 a 389.3. ¿Qué quiere decir esto? Simplemente que los individuos después de la selección y el cruce son mejores que antes de estas transformaciones.

El siguiente paso es volver a realizar la selección y el cruce tomando como población inicial la de la tabla 2.7. Esta manera, de proceder se repite tantas veces, como número de iteraciones se fijen, y ¿cuál es el óptimo? En realidad los algoritmos genéticos no garantizan la obtención del óptimo resultado, sin embargo, si está bien construido, proporcionará una solución razonablemente buena. Pudiendo obtenerse el óptimo, sin embargo, el algoritmo, no confirma que lo sea. Así, ante esto se recomienda, quedarse con la mejor solución de la última iteración. También, es buena idea, ir guardando la mejor solución de todas las iteraciones anteriores, y al final quedarse con la mejor solución de las exploradas.

2.8.1 CONSIDERACIONES PARA ALGORITMOS GENÉTICOS

En problemas reales en los que se aplican los algoritmos genéticos, existe la tendencia a la homogenización de la población, es decir, a que todos los individuos de la misma sean idénticos. Esto impide que el algoritmo siga explorando nuevas soluciones, con lo que se puede quedar estancado en un mínimo local no muy bueno.

Existen técnicas para contrarrestar esta "deriva genética". El mecanismo más elemental, aunque, no sea suficientemente eficaz, es introducir una mutación tras la selección y el cruce. Una vez que, se ha realizado la selección y el cruce se escoge un número determinado de bits de la población y se altera aleatoriamente. En el ejemplo, consiste simplemente en cambiar alguno(s) bit(s) de 1 a 0 ó de 0 a 1.

En conclusión, este capítulo trata de conceptos básicos que comprende un conjunto de conocimientos relacionados sobre los algoritmos de genéticos; esta teoría nos ha permitido conocer la estructura, el comportamiento de los individuos y la población en la aplicación de procesos de tareas en la red, para la optimización de procesos de tareas en el servidor intranet y/o servidor Web.

Los algoritmos genéticos, tienen la capacidad de resolver problemas con un grado de dificultad no muy elevado, con eficiencia y exactitud. Reducen el costo computacional, es decir, el tiempo de cálculo y el consumo de recursos es menor.

Se puede observar la gran ventaja de trabajar con algoritmos genéticos, sobre todo por su sencillez.

La programación mediante algoritmos genéticos supone un nuevo enfoque que permite abarcar todos aquellos campos de aplicación donde no se sabe como resolver un problema, sin embargo, si, se debe ser consciente y poder distinguir que soluciones son buenas y cuales no son buenas.

CAPÍTULO III

3. SISTEMAS DISTRIBUIDOS

3.1 INTRODUCCIÓN

"La computación desde sus inicios ha sufrido muchos cambios, desde las grandes computadoras que permitían realizar tareas en forma limitada, y de uso un tanto exclusivo de organizaciones muy selectas, hasta los actuales computadoras ya sean personales o portátiles que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona" (1).

Los mayores cambios en la evolución de computadores se atribuyen principalmente a dos causas, que se dieron desde las décadas del año de 1970 como:

1. El desarrollo de los microprocesadores, que permitieron reducir en tamaño y costo a las computadoras y aumentar en gran medida las capacidades de los mismos y su acceso a más personas.
2. El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar computadoras con posibilidad de transferencia de datos a alta velocidad.

En este contexto aparece el concepto de "sistemas distribuidos" que se ha popularizado tanto en la actualidad, y que tiene como ámbito de estudio las redes (Hurtado 1997) como: Internet, Internet2, redes de teléfonos móviles, redes corporativas, redes de empresas, redes inalámbricas, etcétera.

Los sistemas distribuidos son sistemas cuyos componentes *hardware* y *software*, se encuentran en computadoras conectadas en red, se comunican y coordinan sus acciones mediante el paso de mensajes / paquetes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente / servidor.

Definición de sistemas distribuidos. Es un conjunto de entidades que se comunican entre si a través de mensajes, los cuales son enviados sobre vías de comunicación (Garcia 1999). Son una colección de elementos de cómputo autónomo que se encuentran físicamente separados y no comparten una memoria común, se comunican entre sí a través del intercambio de mensajes utilizando un medio de comunicación. Los sistemas autónomos pueden tener características no homogéneas.

(1) OMAR HURTADO JARA
Sistemas Distribuidos.

Un sistema distribuido, es aquel en el que dos o más computadoras colaboran para obtención de un resultado. En todo sistema distribuido se establecen una o varias comunicaciones siguiendo un protocolo prefijado mediante un esquema cliente / servidor (Buades 2002).

Un sistema distribuido puede estar conformado por un conjunto de entidades: procesos, computadoras, redes computadoras, dispositivos, procesadores etcétera. Se muestra en la figura 3.1 la conmutación de mensajes / paquetes en LAN siguiente:

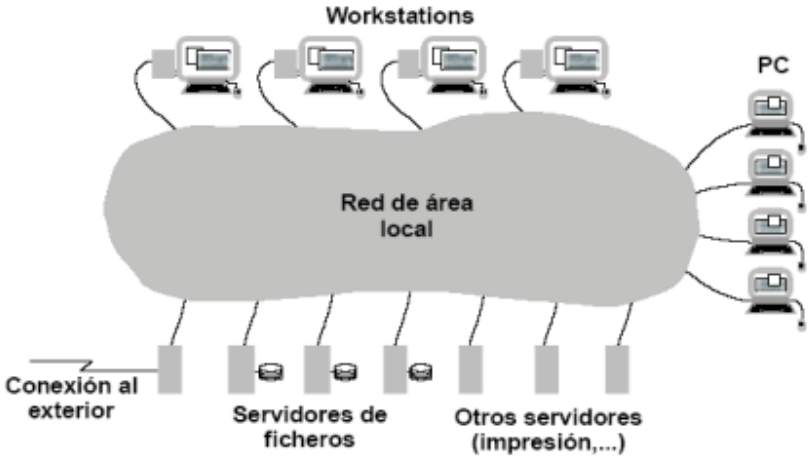


Figura 3.1. Conmutación de mensajes / paquetes en LAN.

3.2 VENTAJAS Y DESVENTAJAS DE SISTEMAS DISTRIBUIDOS

En esta sección, se hace una breve descripción de ventajas y desventajas de sistemas distribuidos (Tanenbaum 1996, Araujo 2004) siguientes:

3.2.1 VENTAJAS

i) Procesadores más poderosos y a menos costo.
- Desarrollo de estaciones con más capacidades.
- Las estaciones satisfacen las necesidades de los usuarios.
- Uso de nuevas interfases.

ii) Avances en la tecnología de comunicaciones.
- Disponibilidad de elementos de comunicación.
- Desarrollo de nuevas técnicas.

iii) Compartición de recursos.
- Dispositivos compartidos (hardware).
- Programas compartidos (software).

iv) Eficiencia y flexibilidad.
- Respuesta rápida del sistema.
- Ejecución concurrente de procesos (en varias computadoras).
- Empleo de técnicas de procesamiento distribuido.

v) Disponibilidad y confiabilidad.
- Sistema poco propenso a fallas, si un componente falla, este no afecta a la disponibilidad del sistema.
- Mayores servicios que elevan la funcionalidad de: monitoreo, telecontrol, correo eléctrico, consultas a la información, etcétera.

vi) Crecimiento modular.
- Es inherente al crecimiento de hardware y software.
- Inclusión rápida de nuevos recursos.
- Los recursos actuales no afectan al sistema.

vii) Los microprocesadores ofrecen mejor proporción precio/rendimiento que los mainframes (economía).

viii) Un sistema distribuido puede tener poder de cómputo que un mainframe (velocidad).

ix) Algunas aplicaciones utilizan máquinas que están separadas a cierta distancia (distribución inherente).

x) Si una máquina se descompone, el sistema puede sobrevivir como un todo (confiabilidad).

xi) Se puede añadir poder de cómputo en pequeños incrementos (crecimiento por incrementos).

3.2.2 DESVENTAJAS

- Requerimientos de mayores controles de procesamiento.
- Velocidad de propagación de información (muy lenta a veces).
- Servicios de replicación de datos y servicios con posibilidades de fallas.
- Mayores controles de acceso y proceso.
- Administración más compleja.
- Costos.
- Interconexión de:
  - Fiabilidad, posible pérdida de mensajes.
  - Posible saturación.
- Algunas comunicaciones inseguras.
- Software más complejo.
- Alguna potencia de cada nodo no adecuada.

3.3 CARACTERÍSTICAS DE SISTEMAS DISTRIBUIDOS

3.3.1 FUNCIONES BÁSICAS DE SISTEMAS DISTRIBUIDOS

Comunicación en interprocesos. La intención es tener comunicación entre procesos sobre la red de conexión, utilizando para ello las facilidades instaladas. Generalmente implica el uso de un protocolo de transporte para establecer el enlace en el ámbito de protocolo de servicios.

Administración y asignación de recursos. Contempla la asignación de recursos a clientes / usuarios, toma de decisión en dónde deberán ser ejecutadas las peticiones, creación e instalación de nuevos recursos en la red, soporte de replicación para procesos críticos, mecanismos de control de concurrencia y sincronización.

Administración de nombres. Mecanismos de asignación y mantenimiento de los nombres de los recursos, localización de servidores / usuarios, mantenimiento de directorios, etcétera.

Reinicio luego de fallas. Implementado en varias capas de operación a lo alto de la arquitectura del sistema.

Funciones de protección. Especificación de los clientes / usuarios y sus derechos de acceso, mecanismos de autentificación, políticas de acceso y contra ataques externos.

Desde el punto de vista del usuario, algunos de los requisitos pueden ser:

- Provee de ayudas para la solución de problemas.
- Minimización del costo de acceso a los recursos.
- Maximización y simplificación de las facilidades de comunicación con otros usuarios o programas, etcétera.

3.4 TIPOS DE SERVIDORES

En esta sección, se hace un breve resumen de tipos de servidores, a continuación se presenta una lista de los servidores más comunes (Garcia 1997, Álvarez 2006), siguientes:

1. **Servidor web**. Red de la Internet / Internet2 que guardan y proporcionan páginas HTML o XML, **protocolo simple de acceso a objetos** (SOAP), para la administración de los sistemas, etcétera. El cliente desde un navegador o enlace puede hacer un llamado de la página y el servidor Web recibe el mensaje y le envía la página correspondiente al cliente que lo solicito.

2. **Servidor de archivos**. Se le ha denominado base de datos ya que proporciona archivos a los clientes / usuarios. Los archivos se utilizan de acuerdo a los requerimientos de los clientes / usuarios y comparten esos archivos entre ellos, es una gran opción de almacenamiento y procesamiento de archivos. El cliente solicita los archivos y el servidor los ubica y lo envía al cliente que lo solicita, etcétera.
3. **Servidor de base de datos**. Es el administrador de base de datos, almacena gran cantidad de datos estructurados y relacionales, se diferencia de los de archivos pues la información que se envía está ya resumida en la base de datos. Por ejemplo, el cliente hace una consulta, el servidor recibe mediante el **lenguaje estructurado de consulta** (SQL) u otro, y extrae solo la información pertinente y envía esa respuesta al cliente.
4. **Servidor de *software* de grupo**. El *software* de grupo es aquel, que permite organizar el trabajo de un grupo. El servidor gestiona los datos que dan soporte a estas tareas. Por ejemplo, para almacenar las listas del correo electrónico. El cliente puede indicar al servidor que se ha terminado una tarea y el servidor lo envía al resto del grupo.
5. **Servidor de correo**. Se encarga de gestionar el envío y recepción de correo electrónico de un grupo de usuarios, el servidor no necesita ser muy potente. El servidor solo debe utilizar un protocolo de correo.
6. **Servidor de objetos**. La mayoría de servidores de este tipo se le considera como el servidor *web* o servidor intranet, permite almacenar objetos que pueden ser activados a distancia. Los clientes pueden ser capaces de activar los objetos que se encuentran en el servidor para procesar sistemas de aplicaciones.
7. **Servidor de impresión**. La gestión de impresión, es administrada por el operador del sistema, que gestionan las solicitudes de impresión de los clientes al servidor. El cliente envía la solicitud de impresión, el servidor recibe la solicitud y la ubica en la cola de impresión, ordena a la impresora que lleve a cabo las impresiones y luego avisa a la computadora cliente que ya acabo su respectiva impresión.
8. **Servidor de aplicaciones**. Es el que realiza las operaciones de procesamiento de aplicaciones empresariales o institucionales. Es básicamente, una aplicación a la que pueden acceder los clientes; etcétera.

3.5 ASPECTOS PRINCIPALES DE SISTEMAS DISTRIBUIDOS

En esta sección, se presenta los aspectos principales de sistemas distribuidos siguientes:

- Tolerancia a fallas.
- Disponibilidad.
- Confiabilidad.
- Sistemas abiertos.
- Concurrencia.
- Escalabilidad.
- Transparencia.

1) TOLERANCIA A FALLAS

El Sistema distribuido puede seguir funcionando, (tal vez con un menor desempeño), a pesar de que uno de sus componentes no este funcionando, como sistemas robustos (Tanenbaum 1996); se tienen los dos enfoques siguientes:

i) Redundancia de *hardware*. Uso de componentes redundantes.

ii) Recuperación de *software*. Diseño de programas para recuperarse de fallas.

2) DISPONIBILIDAD

- La disponibilidad de un sistema, es medida de acuerdo a la proporción del tiempo en el cual esta disponible para su uso.
- La falla en una sola computadora multiusuario, da como resultado la no-disponibilidad del sistema para todos sus usuarios.
- Cuando uno de los componentes falla en un sistema distribuido, solo el trabajo que estaba utilizando el componente es afectado.
- El usuario puede moverse a otra estación si la estación que esta utilizando falla continuamente, o un servidor puede reincializarse en otra computadora.

3) CONFIABILIDAD

- Datos transmitidos a través de vías de comunicación seguras.
- Posibilidad de pérdida y modificación de los datos.

4) SISTEMAS ABIERTOS

- Los Sistemas de información que utilizan estándares comunes para *hardware, software*, aplicaciones y conexiones en la red; para crear un entorno computacional que permite el acceso fácil por parte de los usuarios finales y sus sistemas computacionales conectados a la red.
- En teoría de sistemas, la teoría de los sistemas abiertos es una importante generalización de la teoría física, la cinética y la termodinámica.
- Sistema abierto: determina si el sistema puede extenderse de diferentes formas.
- Los sistemas abiertos se caracterizan por el hecho de que las interfases claves son publicadas.
- Los sistemas abiertos distribuidos proporcionan un mecanismo de comunicación de procesos uniforme y publican interfases para el acceso a recursos compartidos.
- Pueden ser construidos a partir de *software* y *hardware* heterogéneos.

5) CONCURRENCIA

- Varios procesos se encuentran sobre una sola computadora o servidor.
- Ejecución intercalada en el caso de un solo procesador y simultánea si existen n procesadores.
- Ejecución paralela posible debido a:
  - Varios usuarios invocan comandos o interactúan con programas de aplicación.
  - Varios procesos que corren concurrentemente en los servidores.

6) ESCALABILIDAD.

- Los sistemas distribuidos deben operar efectiva y eficientemente en diferentes escalas.
- El Sistema distribuido es práctico si es más pequeño. Por ejemplo dos estaciones y un servidor de archivos.
- Sistemas distribuidos grandes: Es a través de la conexión de varias **interredes** (Internetworks).
- El *Software* de aplicación y el sistema no deben de cambiar cuando la escala del sistema se incremente.
- Ejemplo escalabilidad: El sistema telefónico londinense.

7) TRANSPARENCIA

- Es el ocultamiento al usuario de los programas de ejecución de la disgregación de los componentes de un sistema distribuido.
- **Objetivo**: Es permitir que el sistema sea visto como un todo, más que como una colección de componentes independientes siguientes:
  - Transparencia de acceso.
  - Transparencia de ubicación.
  - Transparencia de concurrencia.
  - Transparencia de replicas.
  - Transparencia de fallas.
  - Transparencia de migración.
  - Transparencia de desempeño.
  - Transparencia de escala.

3.6 SISTEMAS DISTRIBUIDOS Y PARALELOS

En esta sección, se hace un breve resumen de sistemas distribuidos y paralelos, los eventos concurrentes se procesan en forma simultánea en varias computadoras en el servidor *Web* siguiente:

1) Sistemas distribuidos.
   - **Objetivo**: Es compartir recursos y colaborar.
   - Redes de computadoras.
2) Sistemas paralelos.
   - **Objetivo**: Es ejecutar un programa muy rápido (speedup).
   - Máquinas paralelas (arquitecturas dedicadas):
     - Multiprocesadores
     - Multicomputadoras
   - Redes de estaciones de trabajo trabajando como una multicomputadora (cluster).

3.7 COMUNICACIÓN EN LOS SISTEMAS DISTRIBUIDOS

3.7.1 PROTOCOLOS OSI E ISO

Se hace una breve descripción de los protocolos de comunicación, los protocolos son normas estándares que se utilizan para asegurar la compatibilidad de la comunicación entre dos computadoras o más; por ejemplo, los protocolos de Zmódem, TCP/IP, WAP, etcétera. Se muestra en la figura 3.2, comunicación en los sistemas distribuidos, siguiente:

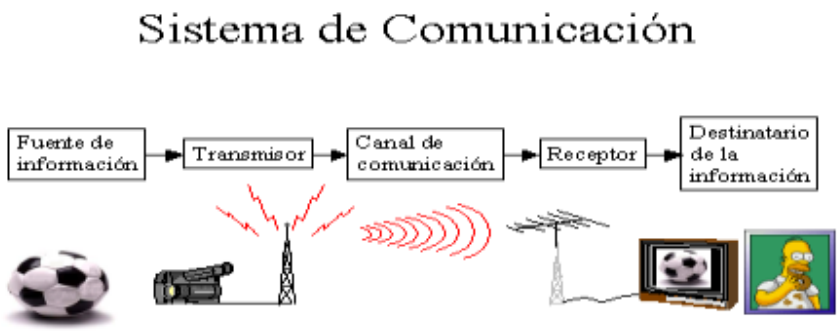


Figura 3.2 Comunicación en los sistemas distribuidos.

3.7.2 PROTOCOLOS UTILIZADOS EN LOS SISTEMAS DISTRIBUIDOS

En esta sección, se hace una breve descripción de los protocolos de los sistemas distribuidos que son los siguientes:

i) **Protocolo de Internet** (IP). Es un protocolo de la capa de red, permite definir la unidad básica de transferencia de datos y se encarga del direccionamiento de la información, para que llegue a su destino en la red.

ii) **Protocolo de Control de Transmisión** (TCP). Es un protocolo de la capa de transporte, permite dividir y ordenar la información a transportar en paquetes de menor tamaño para su transporte y recepción hacia el destino.

iii) **Protocolo de Transferencia de Hipertexto** (HTTP). Es un protocolo de la capa de aplicación, permite el servicio de transferencia de páginas de hipertexto entre el cliente y el servidor Web.
iv) **Protocolo de Transferencia de Correo Simple** (SMTP). Es un protocolo de la capa de aplicación, permite el envío / recepción de correo electrónico por la red.
v) **Protocolo de oficina de correo 3** (POP3). Es un protocolo de la capa de aplicación, permite la gestión de correos en Internet, es decir, le permite a una estación de trabajo recuperar los correos que están almacenados en el servidor.
vi) **Protocolo de acceso inalámbrico** (WAP). Es un estándar mundial establecido para el envío o entrega de información inalámbrica y servicios de telefonía hacia otros terminales inalámbricos, como **asistente personal digital** (PDA), tipo Palm Pilot, Pocket PC, etcétera. Denominado como **protocolo de acceso inalámbrico** (WAP).

3.7.3 MODO DE TRANSMISIÓN ASÍNCRONO

Modo de Transmisión Asíncrono (ATM), es una tecnología de red capaz de transmitir datos, de celdas de alta velocidad utilizada para voz, vídeo y datos. Empaqueta la información en celdas pequeñas y fijas. La celda tiene 53 octetos, de los cuales cinco están reservados para el encabezado de la celda. Es una tecnología orientada a conexión definida por la ITU y el foro ATM. Al nivel más bajo, el ATM envía todos los datos en células fijas con 48 octetos de datos por célula. Un método de transmisión de paquetes utilizando un tamaño de paquete fijo, llamado celda. ATM, es la interfaz de transferencia de datos por RDSI-BA. A diferencia de X.25, ATM, no proporciona mecanismos de control de errores y de control de flujo. Toda la información por transmitir de: voz, datos, imágenes, vídeo se fragmenta primero en tramas pequeñas de tamaño fijo llamadas celdas, las cuales se conmutan y enrutan según los principios de conmutación de paquetes. Esto se denomina también conmutación rápida de paquetes o de celdas. Las primeras redes basadas en este modo de operación fueron las LAN ATM. Que se intercambian entre dos nodos del sistema a velocidades que varían entre 1.5 Mbps y 622 Mbps o superiores (Álvarez 2006).

El modelo cliente/servidor, se basa en la comunicación de simplificación del modelo OSI. Las siete capas que proporciona producen un desaprovechamiento de la velocidad de transferencia de la red, en consecuencia sólo se usarán tres capas:

Capa física (1).
Capa de enlace de datos (2).
Capa de transporte: solicitud / respuesta (5).

Las transferencias se basan en el protocolo de solicitud / respuesta, y se elimina la necesidad de conexión.

Ejemplos de protocolos más utilizados en los sistemas distribuidos:

Protocolo de Internet.
Protocolo de Control de Transmisión.
Protocolo de transferencia de hipertexto.
Protocolo de transferencia de correo simple.
Protocolo de oficina de correo 3.
Protocolo de acceso inalámbrico.

3.7.4 ARQUITECTURA DEL CLIENTE / SERVIDOR

"Una arquitectura es un conjunto de: reglas, definiciones, términos y modelos que se emplean para producir un producto. La arquitectura cliente / servidor agrupa conjuntos de elementos que efectúan procesos distribuidos y computo cooperativo" (1).

Una arquitectura de red esta diseñada sobre el concepto del procesamiento distribuido, donde una tarea se divide entre un servidor, que guarda y distribuye los datos, y un cliente que solicita datos específicos al servidor. Es un modelo de red, en la que toda la información es proporcionada y administrada por una sola computadora llamada servidor. Este modelo comparte los recursos de software: sistema operativo, sistemas de aplicaciones; realizan los procesos de trabajos y retorna los resultados a los usuarios. La distribución de sistemas en diferentes nodos y la forma de como se comunican entre sí, se muestra en la figura 3.4 siguiente:

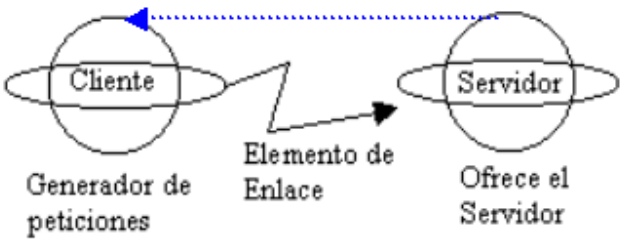


Figura 3.3. Distribución de sistemas en diferentes nodos y como se comunican entre sí.

(1) Alonso Araujo Cárdenas
Sistemas Distribuidos, Conceptos y Características.

3.7.4.1 BENEFICIOS

- Mejor aprovechamiento de la potencia de cómputo (distribuye el trabajo).
- Reduce el tráfico en la red (viajan requerimientos).
- Opera bajo sistemas abiertos.
- Permite el uso de interfases gráficas variadas y versátiles, etcétera.

3.7.4.2 CLIENTE

Conjunto de hardware y software que invoca los servicios de uno o varios servidores.

a) CARACTERÍSTICAS DEL CLIENTE

- El cliente oculta al servidor y la red.
- Detecta e intercepta peticiones de otras aplicaciones y pueden redireccionarlas.
- Dedicado a la sesión del usuario (inicia-termina).
- El método más común por el que se solicitan los servicios es a través de llamadas de procedimientos remotos (RPC).

b) FUNCIONES COMUNES DEL CLIENTE

- Mantener y procesar todo él dialogo con el usuario.
- Manejo de pantallas.
- Menús e interpretación de comandos.
- Entrada de datos y validación.
- Procesamiento de ayudas.
- Recuperación de errores.

3.7.4.3 SERVIDOR

Conjunto de hardware y software que responden a los requerimientos de un cliente.

i) Tipos comunes de servidores son:

- Servidor de archivos (FTP, Novell).
- Servidor de bases de datos (SQL, CBASE, Oracle, INFORMIX, etc.).
- Servidor de comunicaciones.
- Servidor de impresión.
- Servidor de terminal.
- Servidor de aplicaciones (servidor Windows 2003, Linux, Novell, IBM, etcétera).

ii) Funciones comunes del servidor son:

- Acceso, almacenamiento y organización de datos.
- Actualización de datos almacenados.
- Administración de recursos compartidos.
- Ejecución de toda la lógica para procesar una transacción.
- Procesamiento común de elementos del servidor.

3.7.5 LLAMADA A PROCEDIMIENTO REMOTO

Llamada a procedimiento remoto (RPC). Fue creado por Birell y Nelson en 1984, es un procedimiento que realiza la llamada remota y el que la recibe se ejecuta en máquinas diferentes, utilizan espacios de direcciones distintos para dichas tareas (Francisco Dueñas 2003).

Así mismo, se deben transferir los parámetros y los resultados a las máquinas solicitantes. Los trabajos no se ejecutan cuando las máquinas no son idénticas o no hay protocolos de comunicación. Por excepción, se pueden descomponer ambas máquinas y cada una de las posibles fallas puede ser la causa de diversos problemas. El sistema operativo es una técnica por la que dos programas de máquinas diferentes pueden interactuar utilizando la sintaxis y la semántica de llamadas y retornos de los procedimientos. Tanto el programa llamado como el llamador se comportan como si el programa asociado estuviera ejecutando en la misma máquina.

El diseño de un sistema operativo distribuido plantea la **llamada a procedimiento remoto** (RPC). Los RPC amplían la llamada local a procedimientos, y los generalizan a una llamada a un procedimiento localizado en cualquier lugar de todo el sistema distribuido. En un sistema distribuido no se debería distinguir entre llamadas locales y RPC, lo que favorece en gran medida la transparencia del sistema.

Una de las dificultades más evidentes a las que se enfrenta el RPC es el formato de los parámetros de los procedimientos. Un ejemplo, para ilustrar este problema es la posibilidad de que en un sistema distribuido formado por diferentes tipos de computadoras, una computadora con formato little endian llamara a un procedimiento de otra computadora con formato big endian, etcétera. Este problema se podría solucionar si se tiene en cuenta que ambos programas conocen el tipo de datos de los parámetros, o estableciendo un estándar en el formato de los parámetros, de forma que sea utilizado de forma única.

Otro problema de peor solución es el paso de apuntadores como parámetros. Debido a que los apuntadores guardan una dirección del espacio de direcciones local, el procedimiento que recibe el apuntador como parámetro no puede utilizar inmediatamente el apuntador, ya que no tiene acceso a los datos, que para él son remotos. En el futuro se utilizará memoria compartida, que propone una solución a este problema.

Por último queda por solucionar la tolerancia a fallos. Una llamada a un procedimiento remoto puede fallar por motivos que antes no existían, como la pérdida de mensajes o el fallo del cliente o del servidor durante la ejecución del procedimiento.

La limitación del RPC más clara en los sistemas distribuidos es que no permite enviar una solicitud y recibir respuesta de varias fuentes a la vez, sino que la comunicación se realiza únicamente entre dos procesos. Por motivos de tolerancia a fallos, bloqueos, u otros, sería interesante poder tratar la comunicación en grupo. Por ejemplo, se muestra en la figura 3.4 comunicaciones de sistemas operativos distribuidos siguientes:

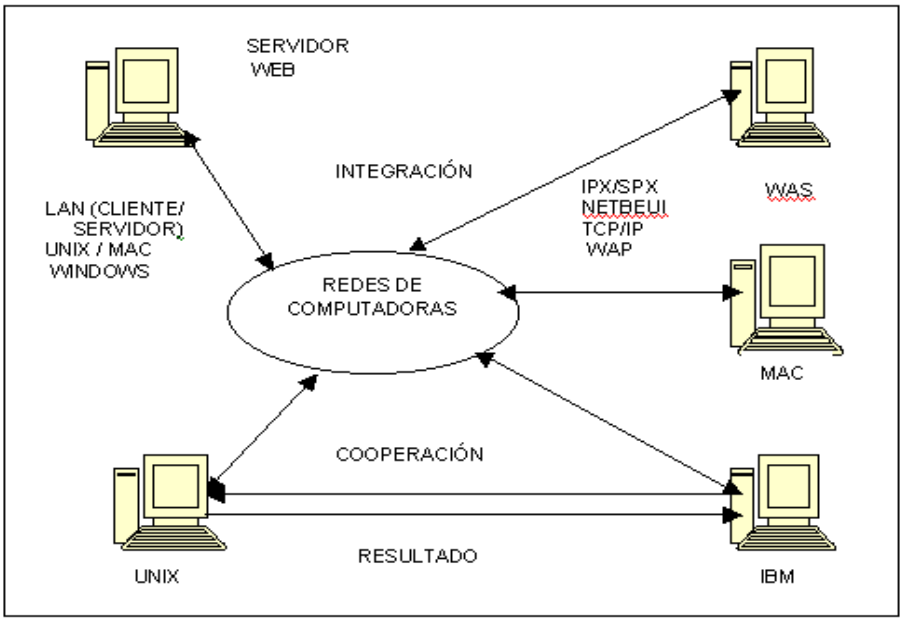


Figura 3.4. Comunicaciones de sistemas operativos distribuidos.

3.8 OBJETOS DISTRIBUIDOS

En los sistemas cliente / servidor, un objeto distribuido es aquel que esta gestionado por un servidor y sus clientes e invocan sus métodos utilizando un "**método de invocación remota**". El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje.

3.8.1 TECNOLOGÍAS ORIENTADAS A OBJETOS DISTRIBUIDOS

Las tres tecnologías importantes y más utilizadas en este ámbito son:

1. **Método de Invocación Remota** (RMI). Fue el primer marco o estructura para crear sistemas distribuidos en Java. El método de la invocación remota de Java que permite llamar a un objeto que se está ejecutando en una **Máquina Virtual** (VM), Java llama al método de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.

2. **Modelo de objeto de componente distribuido** (DCOM). El modelo de objeto de componente distribuido, esta incluido en los sistemas operativos de Microsoft. Es una representación de conceptos e interfases del programa, en el cual los objetos de programa del cliente, pueden solicitar servicios de objetos de programa servidores en otras computadoras dentro de una red. Esta tecnología esta asociada a la plataforma de productos de Microsoft.

3. **Arquitectura de agente de solicitud de objetos comunes** (CORBA). Tecnología que fue introducido por el **grupo de administración de objetos** (OMG), creada para establecer una plataforma para la gestión de objetos remotos independientes del lenguaje de programación.

3.9 DESARROLLO WEB

Es un caso particular de los sistemas cliente / servidor con representación remota. En donde se dispone de un protocolo estándar: HTTP y un Middleware denominado **servidor web** (WebServer). En la actualidad la aplicación de sistemas informáticos basados en la Internet, es una herramienta fundamental para las organizaciones que desean tener cierta presencia competitiva.

3.9.1 TECNOLOGÍAS LÓGICAS DE APLICACIÓN EN SERVIDOR WEB

En esta sección, se hace una breve descripción de las tecnologías de la lógica de la aplicación en servidor web (Buades 2002), siguientes:

a) **Interfase Común de Pasarela** (CGI). Son programas que se ejecutan en el servidor, pueden servir como pasarela con una aplicación o base de datos o para generar documentos HTML o XML de forma automática. Cada petición HTTP ejecuta un proceso, el cual analiza la solicitud y genera un resultado. Son independientes del sistema operativo, y presentan la ventaja de que, dado un programa escrito en un lenguaje cualquiera, es fácil adaptarlo a un CGI. Entre los lenguajes que se utilizan para CGIs, el más popular es el Perl y otros.

b) **Servlets.** Pequeños programas en Java que se ejecutan de forma persistente en el servidor, y que, por lo tanto, tienen una activación muy rápida, y una forma más simple de hacerlo. Estos programas procesan una petición y generan la página de respuesta.

c) **Página de Servidor Activo** (ASP). Una página ASP es un archivo (o fichero) de sólo texto que contiene las secuencias de comandos, junto con el HTML o XML necesarios, y que se guarda con la extensión ".asp".

Al ser llamado por el navegador, el motor ASP del **servidor de información de Internet** (IIS) se encarga automáticamente de ejecutarlo como se suele hacer con un programa cualquiera, sin embargo, cuya salida siempre será a través del navegador que le invoca. Es un entorno propietario de Microsoft y el lenguaje de secuencia de comandos predeterminado del IIS es el VBScript, aunque puede cambiarse.

d) **Servidor de Páginas de Java** (JSP). Servidor de páginas de Java que consisten en pequeños trozos de código en Java que se insertan dentro de páginas web, de forma análoga a los ASPs. Ambas opciones, hoy en día, son muy populares en sitios de comercio electrónico. Frente a los ASPs, la ventaja que presentan es que son independientes del sistema operativo y del procesador de la máquina.

e) **Pre Procesador Hipertexto** (PHP). Es un lenguaje cuyos programas se insertan también dentro de las páginas web, al igual que los ASPs y JSPs; es mucho más simple de usar, y el acceso a bases de datos desde él es muy simple. Es más utilizado en sitios de comercio electrónico con poco tráfico, por su facilidad de desarrollo y rapidez de implementación.

f) Las consideraciones a tomar en el desarrollo de unos sistemas web, siguientes:

- Separar la lógica de la aplicación de la interfase del usuario.
- Utilizar métodos estándar de comunicación entre la lógica de aplicación y la interfase del usuario.
- Herramientas que permitan una fácil adaptación de las aplicaciones a los nuevos dispositivos que irán apareciendo.
- Definir el costo en comunicaciones que debe asumir la organización.
- Tener en cuenta los procesos de réplica, periodicidad y el ancho de banda que consuman.
- Replantear la idoneidad de la ubicación de cada proceso.
- Extremar las pruebas al diseñar e implementar los protocolos de comunicación.

g) Las Tendencias actuales de las arquitecturas de sistemas web, se muestra en la figura 3.5 siguiente:

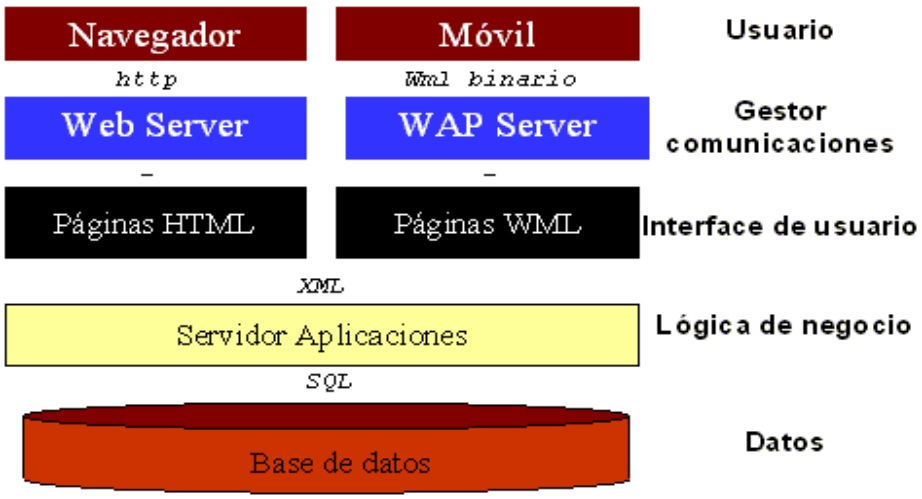


Figura 3.5. Tendencias actuales de las arquitecturas de sistemas *web.*

3.10 TECNOLOGÍAS INALÁMBRICAS

Las tecnologías inalámbricas, en los últimos años, están alcanzando la madurez necesaria para permitir el acceso a una red, sin la necesidad de la utilización de los cables tradicionales de conexión (Hurtado 1997).

A continuación se presenta un conjunto de tecnologías que contribuyen al desarrollo de las conexiones inalámbricas:

a) SISTEMA GLOBAL PARA COMUNICACIONES MÓVILES

El sistema global para comunicaciones móviles (GSMC), es un estándar para comunicación que utilizan los teléfonos móviles que incorpora la tecnología digital. Permite utilizar el sistema de **servicio de mensajes cortos** (SMS), para enviar y recibir mensajes de texto. Es la evolución tecnológica de los teléfonos móviles análogos.

b) SERVICIO DE RADIO DE PAQUETE EN GENERAL

Servicio de radio de paquete en general (GPRS). Es un sistema de transmisión que funciona en el entorno de la telefonía móvil. En este sistema cada llamada de voz o cada conexión de datos, ocupa de manera exclusiva un canal mientras dura esa llamada o conexión, por tanto, un usuario puede hacer uso de varios canales y un mismo canal puede ser compartido por varios usuarios. Esta basado en la conmutación de paquetes y permite la transmisión de datos a alta velocidad para el acceso a Internet.

c) SISTEMAS TELECOMUNICACIONES MÓVILES DEL UNIVERSO

Sistemas telecomunicaciones móviles del universo (UMTS). El sistema universal de telecomunicaciones móviles, permitirá disponer de banda ancha en telefonía móvil y transmitir un volumen de datos importante por la red. Con esta tecnología de tercera generación serán posibles las videoconferencias, descargar videos, el intercambio de postales electrónicas, paseos “virtuales” por casas en venta, etcétera, todo desde el móvil.

d) PROTOCOLO DE APLICACIÓN INALÁMBRICA

El **protocolo de aplicación inalámbrica** (WAP), es un servicio de mensajes digital inteligente para teléfonos celulares y otras terminales móviles que les permitirán visualizar contenidos de la Internet en un formato de texto especial en un teléfono celular con tecnología del **sistema global para comunicaciones** (GSM).

WAP se ha convertido en el estándar global para proveer información a las terminales inalámbricas.

WAP utiliza un micro navegador con un nuevo estándar llamado **lenguaje de marcas inalámbricas** (WML) (similar al HTML) optimizado para terminales móviles inalámbricas.

WAP esconde la complejidad del GSM en las aplicaciones, así como la web lo ha hecho para Internet. Expande una variedad de opciones de transporte y dispositivos, incluyendo SMS, 9.6 kbps GSM data y GPRS.

e) BLUETOOTH

Es la norma que define un estándar global de comunicación inalámbrica a cortas distancias, que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia. Los principales objetivos que se pretende conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

La tecnología Bluetooth comprende *hardware, software* y requerimientos de interoperatividad.

f) FIDELIDAD INALÁMBRICA

Fidelidad inalámbrica (WI-FI). Es la tecnología utilizada en una red o conexión inalámbrica, para la comunicación de datos entre equipos situados dentro de una misma área (interior o exterior) de cobertura.

Conceptualmente, no existe ninguna diferencia entre una red con cables (cable coaxial, fibra óptica, etc.) y una inalámbrica. La diferencia está en que las redes inalámbricas transmiten / reciben datos a través de ondas electromagnéticas, lo que supone la eliminación del uso de cables y, por tanto, una total flexibilidad en las comunicaciones.

g) INTEROPERATIVIDAD MUNDIAL PARA EL ACCESO A MICROONDA.

Interoperatividad mundial para el acceso a microonda (WIMAX). Es el nombre con el que se conoce la norma 802.16e, es un estándar inalámbrico aprobado en enero de 2003, en el Foro WIMAX, formado por un grupo de 67 compañías, que ofrece un mayor ancho de banda y alcance que la familia de estándares WiFi, compuesta por el 802.11a, 802.11b y 802.11g. Denominado como **interoperatividad mundial para el acceso a microonda** (WIMAX). Se determina la diferencia entre estas dos tecnologías inalámbricas que son alcance y ancho de banda. Mientras que WiFi está determinada para oficinas o dar cobertura a zonas relativamente pequeñas, WiMax ofrece las tasas de transferencias de 70 Mbps a distancias hasta 50 kilómetros de una estación base. En comparación, la tasa de transferencia de WiFi es de 11 Mbps y la distancia de hasta 350 metros en zonas abiertas.

3.11 DESAFÍOS

Heterogeneidad de los componentes. La interconexión, sobre todo cuando se utiliza Internet, se da sobre una gran variedad de elementos *hardware* y *software*, no necesariamente compatibles por lo cual necesitan de ciertos estándares que permitan esta comunicación. Los Middleware, son elementos software que permiten una abstracción de la programación y el enmascaramiento de la heterogeneidad subyacente sobre las redes. También el middleware proporciona un modelo computacional uniforme.

Extensibilidad. Determina si el sistema puede extenderse y reimplementarse en diversos aspectos (añadir y quitar componentes). La integración de componentes escritos por diferentes programadores que auténtico reto.
Seguridad. Reviste gran importancia por el valor intrínseco para los usuarios. Tiene tres componentes siguientes:

Confidencialidad. Protección contra individuos no autorizados.
Integridad. Protección contra la alteración o corrupción.
Disponibilidad. Protección contra la interferencia con los procedimientos de acceso a los recursos.

Escalabilidad. El sistema es escalable si conserva su efectividad al ocurrir un incremento considerable en el número de recursos y en el número de usuarios.
Tratamiento de fallos. La posibilidad que tiene el sistema para seguir funcionando ante fallos de algún componente en forma independiente, sin embargo, para esto se tiene que tener alguna alternativa de solución. Técnicas para tratar fallos siguientes:

Detección de fallos. Por ejemplo, algunos fallos son detectables, con comprobaciones.

Enmascaramiento de fallos. Algunos fallos detectados pueden ocultarse o atenuarse.

Tolerancia a fallos. Sobre todo en Internet se dan muchos fallos y no es muy conveniente ocultarlos, es mejor tolerarlos y continuar. Por ejemplo, tiempo de vida de una búsqueda.

Recuperación frente a fallos. Tras un fallo se deberá tener la capacidad de volver a un estado anterior.

Redundancia. Se puede utilizar para tolerar ciertos fallos (DNS, BD, etc.).

Concurrencia. Compartir recursos por parte de los clientes a la vez.
Transparencia. Es la ocultación al usuario y al programador de aplicaciones de la separación de los componentes en un sistema distribuido. Se identifican ocho formas de transparencias siguientes:

De acceso. Se accede a recursos locales y remotos de forma idéntica.

De ubicación. Permite acceder a los recursos sin conocer su ubicación.

De concurrencia. Utilizar un recurso compartido sin interferencia.

De replicación. Permite utilizar varios ejemplares de cada recurso.

Frente a fallos. Permite ocultar los fallos.

De movilidad. Permite la reubicación de recursos y clientes sin afectar al sistema.

De prestaciones. Permite reconfigurar el sistema para mejorar las prestaciones según su carga.

De escalado. Permite al sistema y a las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.

3.12 APLICACIONES

a) **Sistemas comerciales**

Inicialmente fueron, construidos con hardware dedicado y entornos centralizados, son características de distribución geográfica y necesidad de acceso a sistemas distintos, ideales para implementarse en sistemas distribuidos. Requieren ciertas características de fiabilidad, seguridad y protección. Algunos ejemplos son:

- Sistemas de reservas de líneas aéreas.
- Aplicaciones bancarias.
- Cajas y gestión de grandes almacenes.

b) **Red de área extensa / amplia** (WAN)

Debido al gran crecimiento de este tipo de redes (Internet), ha tomado gran importancia el intercambio de información a través de la red y para ello se tiene los ejemplos siguientes:

- Los servicios comunes que brinda Internet: correo electrónico, servicio de noticias, transferencia de archivos, la WWW, etcétera.

Aplicaciones multimedia. Son las últimas incorporaciones a los sistemas distribuidos. Estas aplicaciones imponen ciertas necesidades de hardware para poder tener una velocidad y regularidad de transferencia de una gran cantidad de datos. Los ejemplos de estos sistemas son:

- Videoconferencia.
- Televigilancia.
- Juegos multiusuarios.
- Enseñanza asistida por la computadora.

Áreas de la informática aplicada a los sistemas distribuidos. En este punto se tienen en cuenta toda la variedad de aplicaciones de los sistemas distribuidos, pues su diseño involucra a varias áreas como:

- Comunicaciones.
- Sistemas operativos distribuidos.
- Base de datos distribuidos.
- Servidores distribuidos de archivos / ficheros.
- Lenguajes de programación distribuidos.
- Sistemas de tolerancia a fallos.

3.13 RED DE COMUNICACIÓN

Es todo aquel conjunto de elementos basados en *hardware* y *software* que permite establecer un enlace entre los clientes y los servidores de una red de computadoras, se clasifican por su tamaño LAN, MAN y WAN.

Las características de las comunicaciones son:

- A través de este medio, el cliente debe localizar e iniciar la comunicación con el servidor.
- No se utiliza la metodología de compartición de archivos, ya que todos los accesos a la información se llevan a cabo a través de peticiones por medio de comunicación.
- Debido a que los programas de manejo y control de información de: archivos y bases de datos solo se envían y reciben, los resultados de las operaciones del tráfico igual a datos leídos o escritos.

- Debido a la flexibilidad de establecer sesiones con múltiples servidores y manejo de información en varias bases de datos, como en sitios remotos es requerido el uso de estilos transaccionales y cooperativos.

3.14 PROCESOS EN LOS SISTEMAS DISTRIBUIDOS

3.14.1 MEMORIA CACHÉ

Con el aumento de la rapidez de los microprocesadores ocurrió la paradoja de que las memorias principales no eran suficientemente rápidas como para poder ofrecerles a los datos que éstos necesitaban. Por esta razón, las computadoras comenzaron a construirse con una memoria caché interna situada entre el microprocesador y la memoria principal. La memoria caché permite incrementar la velocidad de transmisión de datos, en el almacenamiento temporal de datos y mejora del procesamiento en la CPU.

Existen tres tipos de memoria caché, se diferencian por ubicación y funcionamiento, son:

L1 Interna. Situada dentro del propio procesador y, por tanto, de acceso aún más rápido y aún más cara. La caché de primer nivel contiene muy pocos kilooctetos (unos 32 ó 64 Kb).

L2 Externa. Situada entre el procesador y la RAM. Los tamaños típicos de la memoria caché L2 oscilan en la actualidad entre 256 Kb y 4 Mb: la memoria caché es un tipo especial de memoria que poseen las computadoras. Esta memoria se sitúa entre el microprocesador y la memoria RAM, y se utiliza para almacenar datos que se utilizan frecuentemente. Permite agilizar la transmisión de datos entre el microprocesador y la memoria principal. Es de acceso aleatorio (también conocida como acceso directo) y funciona de forma similar a como lo hace la memoria principal (RAM), aunque es mucho más rápida.

L3. Esta memoria se encuentra por lo general en las placas base.

i) CACHÉ WEB

Una caché web denominado servidor Proxy, es una entidad de red que satisface las peticiones HTTP en nombre de un servidor original. La caché web tiene su propio almacenamiento en disco, y en el guarda las copias de los objetos que más recientemente han sido pedidos. Como se muestra en la figura 3.6, un navegador de usuario puede ser configurado de forma que todas las peticiones HTTP del usuario sean dirigidas en primer lugar a la caché web. Por ejemplo, éste es un procedimiento directo con los navegadores de Internet Explorer y Netscape. Una vez que el navegador ha sido configurado, cada petición del navegador de un objeto se dirige en primer lugar a la caché web. Por ejemplo, supongamos que un navegador que pide el objeto http://www.escuela.edu/campus.gif (Kurose y Ross 2003). Esto es lo que ocurren en:

1. El navegador establece una conexión TCP con la caché Web y envía una petición HTTP del objeto a la *caché web*.

2. La caché web comprueba si tiene una copia del objeto almacenada localmente. Si es así, envía el objeto al navegador cliente dentro de un mensaje HTTP de respuesta.
3. Si la caché *Web* no tiene el objeto, abre una conexión TCP con el servidor web A, esto es, con www.escuela.edu. La caché *web* envía entonces una petición HTTP del objeto sobre la conexión TCP. Después de recibir está petición, el servidor web A envía el objeto a caché web dentro de una respuesta HTTP.
4. Cuando la caché Web recibe el objeto, almacena una copia en su almacenamiento local y envía una copia, dentro de un mensaje HTTP de respuesta, al navegador cliente (sobre la conexión TCP existente entre el navegador cliente y caché web).

Obsérvese que una caché es al mismo tiempo un servidor y cliente. Cuando recibe peticiones y envía respuestas a un navegador, es un servidor. Cuando envía peticiones y recibe respuestas de un servidor web B, es un cliente.

Típicamente, la caché web es comprobada e instalada por un ISP. Por ejemplo, una universidad puede instalar una caché en su red de campus, y configurar todos los navegadores del campus referenciando a caché. O un ISP residencial importante (como AOL), podría instalar una o más cachés en su red, y preconfigurar los navegadores enviados referenciados a caché Web que están instalados.

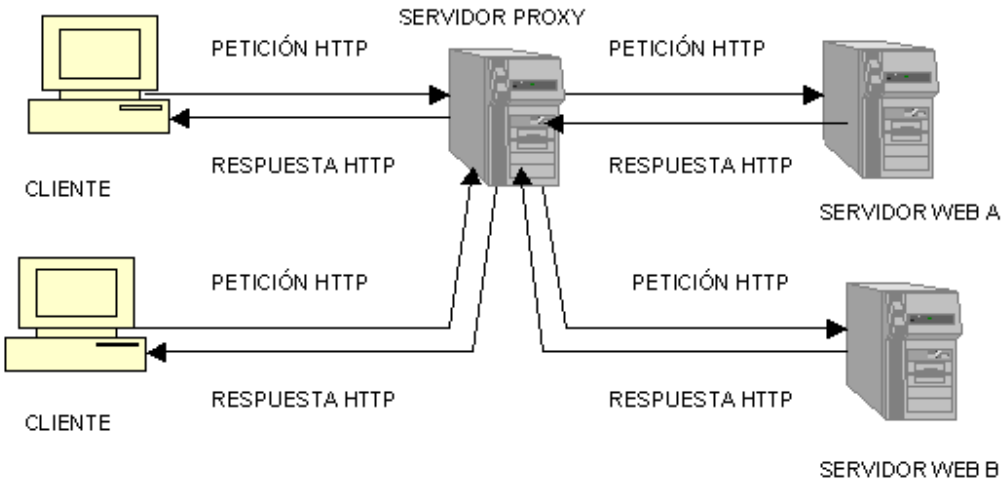


Figura 3.6. Clientes pidiendo objetos a través de una caché web.

La caché web es una forma de distribución de contenidos, ya que replica el contenido en caché y proporciona a los usuarios un mecanismo para acceder contenido replicado. Obsérvese que el proveedor de contenido no supervisa la replicación; al contrario, el contenido es replicado a petición como una función de las peticiones del usuario.

La caché web se ha desarrollado en Internet por las tres razones siguientes:

Primero. Una caché web puede reducir sustancialmente el tiempo de respuesta de una petición de cliente, particularmente si el ancho de banda entre el cliente y el servidor web A es mucho menor que entre el cliente y caché. Si existe una conexión de alta velocidad entre el cliente y caché, como ocurre habitualmente, y si caché tiene el objeto pedido, entonces, la caché podría entregar rápidamente el objeto al cliente.

Segundo. Como se verá en un ejemplo, las cachés web pueden reducir sustancialmente el tráfico en el enlace de acceso a Internet de una institución. Al reducir el tráfico, la institución (por ejemplo, una empresa o una universidad) no tiene que mejorar el ancho de banda tan rápido y, por tanto, se reducen los costos. Es más, las cachés web pueden reducir sustancialmente el tráfico global en Internet, con lo que sé mejora el rendimiento de todas las aplicaciones.

Tercero. Una Internet poblada de cachés web (en el ámbito institucional, regional nacional) proporciona una infraestructura para la distribución rápida de contenidos, incluso para proveedores que tienen sus sitios en servidores de baja velocidad sobre enlaces de acceso de baja velocidad. Si este proveedor de contenido es pobre de recursos adquiere de repente popularidad en los contenidos que distribuye, estos contenidos se copiarán rápidamente él las cachés de Internet, y la alta demanda de los clientes / usuarios será satisfecha.

CAPÍTULO IV

4. BASES DE DATOS DISTRIBUIDOS

4.1 INTRODUCCIÓN

Base de Datos es una colección de datos estructurados sobre una red y que pertenecen, lógicamente, a un solo sistema distribuido (Omar Hurtado 1997), la cual cumple las condiciones siguientes:

- La información de la base de datos esta almacenada físicamente en diferentes sitios de la red.
- En cada sitio de la red, la parte de la información, se constituye como una base de datos en sí misma.
- Las bases de datos locales tienen sus propios usuarios locales, su propio **Sistema de Gestión de Bases de Datos** (DBMS) y programas para la administración de transacciones, y su propio administrador local de comunicación de datos.
- Estas bases de datos locales deben de tener una extensión, que gestione las funciones de sociedad necesarias; la combinación de estos componentes con los sistemas de administración de base de datos locales, es lo que se conoce como sistema administrador de base de datos distribuidos.
- Este gestor global permite que los usuarios puedan acceder a los datos desde cualquier punto de la red, como si lo hicieran con los datos de su base de datos local, es decir, para el usuario, no debe existir diferencia entre trabajar con datos locales o datos de otros sitios (remotos) de la red.

Por lo expuesto, una base de datos distribuidos es como una unidad virtual, cuyas partes se almacenan físicamente en varias bases de datos "reales" distintas, ubicadas en diferentes sitios.

Base de Datos Distribuidos. En sistemas distribuidos la base de datos es implementada en una red de computadoras. Las particiones se distribuyen sobre varias estaciones de trabajo de la red. Dependiendo de la actualización específica de la información y del tráfico de recuperación y distribución de base de datos puede aumentar significativamente el rendimiento general. El hecho de poder compartir datos se hace posible a través de una red que interconecta las bases de datos distribuidos. Las bases de datos trabajan con datos distribuidos en una red o en varias computadoras diferentes, una característica esencial de una base de datos distribuidos es que el usuario y/o programa funciona como sí tuvieran acceso local a toda la base de datos. Todo el proceso se lleva a cabo mediante un sistema de gestión de base de datos (Hansen 1997).

A diferencia de los sistemas paralelos, en los que los procesadores se hallan estrechamente acoplados y constituyen un solo sistema de bases de datos, los siste-

mas de bases de datos distribuidos están formados por sitios débilmente acoplados que no comparten ningún componente físico. Además, podría ser que los sistemas de bases de datos que se ejecutan en cada sitio sean sustancialmente independientes entre sí.

Más adelante se describe el control de concurrencia en las bases de datos distribuidos. Además, se esboza el modo de proporcionar una elevada disponibilidad en bases de datos distribuidos aprovechando las réplicas, de manera que el sistema pueda continuar procesando las transacciones aunque se produzca una avería. El procesamiento de consultas en las bases de datos distribuidos. También, se tratarán algunos aspectos del manejo de bases de datos heterogéneas, y los sistemas de directorio, que pueden considerarse de una forma especializada de las bases de datos distribuidos (Silberschatz, Korth y Sudarshan 2006).

4.2 EJEMPLOS DE BASE DE DATOS DISTRIBUIDOS

Supongamos, un banco que tiene tres sucursales, en cada sucursal, hay una computadora que controla las terminales de la misma y el sistema de cuentas. Cada computadora con su sistema de cuentas local en cada sucursal constituye un "sitio" de la base de datos distribuidos; las computadoras están conectadas por la red. Durante las operaciones normales, las aplicaciones en las terminales de la sucursal necesitan sólo acceder la base de datos de la misma. Como sólo acceden a la misma red local, se les llaman aplicaciones locales.

Desde el punto de vista tecnológico, aparentemente lo importante es la existencia de algunas transacciones que acceden a información en más de una sucursal. Estas transacciones son llamadas transacciones globales o transacciones distribuidos.

La existencia de transacciones globales será considerada como una característica que nos ayude a discriminar entre las **Bases de Datos Distribuidos** (BDD) y un conjunto de base de datos locales.

Una típica transacción global sería una transferencia de fondos de una sucursal a otra. Esta aplicación requiere de actualizar datos en dos diferentes sucursales y asegurarse de la real actualización en ambos sitios o en ninguno. Asegurar el buen funcionamiento de aplicaciones globales es una tarea difícil. Más adelante se muestra en la figura 4.1 bases de datos distribuidos homogéneos, siguiente:

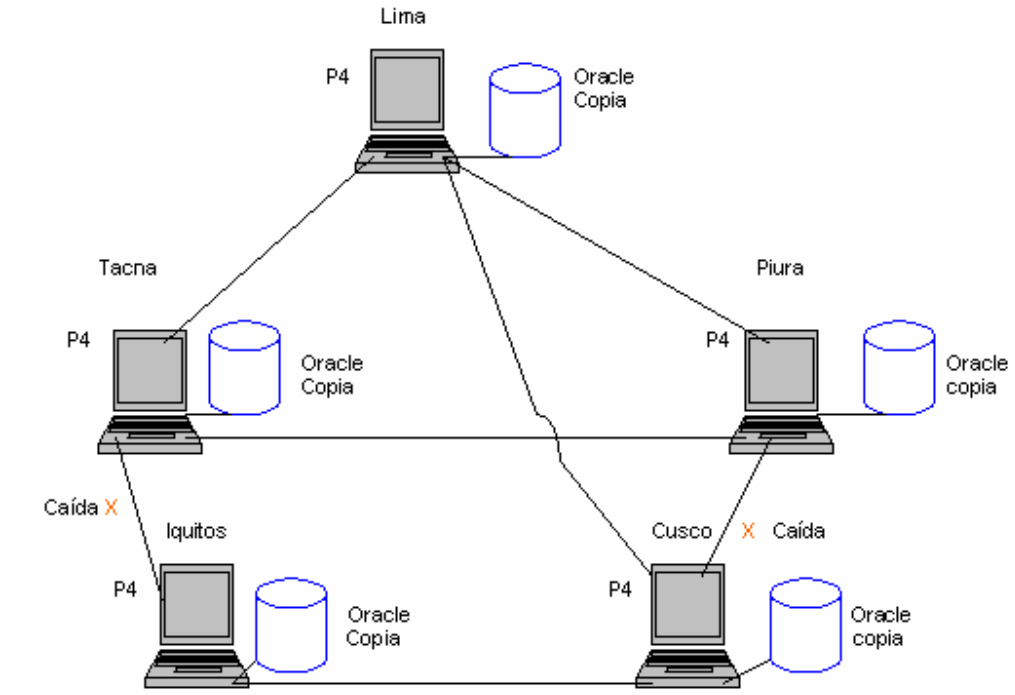


Figura 4.1 Bases de datos distribuidos homogéneos.

4.3 VENTAJAS Y DESVENTAJAS DE BASE DE DATOS DISTRIBUIDOS

4.3.1 VENTAJAS DE BASE DE DATOS DISTRIBUIDOS

Descentralización. En un sistema centralizado / distribuido, existe un administrador que controla toda la base de datos, por el contrario en un sistema distribuido existe un administrador global que lleva una política general y delega algunas funciones a administradores de cada localidad para que establezcan políticas locales y así, realizar un trabajo eficiente.

Economía. Existen dos aspectos a tener en cuenta:

- El primer aspecto, son los costos de comunicación; si las bases de datos están muy dispersas y las aplicaciones hacen amplio uso de los datos puede resultar más económico dividir la aplicación y realizarla localmente.
- El segundo aspecto, es que cuesta menos crear un sistema de pequeñas computadoras con la misma potencia que una única computadora.

Mejora el rendimiento. Pues los datos serán almacenados y utilizados donde son generados, lo cual permitirá distribuir la complejidad del sistema en los diferentes sitios de la red, optimizando la labor.

Mejora la fiabilidad y disponibilidad. La falla de uno o varios lugares o el de un enlace de comunicación no implica la inoperatividad total del sistema, incluso si tenemos datos duplicados puede que exista una disponibilidad total de los servicios.

Mejor crecimiento. Es más fácil acomodar el incremento del tamaño en un sistema distribuido, por que la expansión se lleva a cabo añadiendo poder de procesamiento y almacenamiento en la red, al añadir un nuevo nodo.
Flexibilidad. Permite acceso local y remoto de forma transparente.
Disponibilidad. Pueden estar los datos duplicados con lo que varias personas pueden acceder simultáneamente de forma eficiente. El inconveniente, el sistema administrador de base de datos debe preocuparse de la consistencia de los mismos.
Control de concurrencia. El sistema administrador de base de datos local se encarga de manejar la concurrencia de manera eficiente.

4.3.2 DESVENTAJAS DE BASE DE DATOS DISTRIBUIDOS.

- El rendimiento que es una ventaja podría verse contradicho, por la naturaleza de la carga de trabajo, pues un nodo puede verse abrumado, por las estrategias utilizadas de concurrencia y de fallos, y el acceso local a los datos. Se puede dar esta situación cuando la carga de trabajo requiere un gran número de actualizaciones concurrentes sobre datos duplicados y que deben estar distribuidos.
- La confiabilidad de los sistemas distribuidos, esta entre dicha, puesto que, en este tipo de base de datos existen muchos factores a tomar en cuenta como: la confiabilidad de las computadoras de la red, del sistema de gestión de base de datos distribuidos de las transacciones y de las tasas de error de la carga de trabajo.
- La mayor complejidad, representa en contra de este tipo de sistemas, pues muchas veces se traduce en altos gastos de construcción y mantenimiento. Esto se da por la gran cantidad de componentes hardware, muchas cosas que aprender, y muchas aplicaciones susceptibles de fallar. Por ejemplo, el control de concurrencia y recuperación de fallos, requiere de personal muy especializado y por tal motivo es costoso.
- El procesamiento de base de datos distribuidos es difícil de controlar, pues estos procesos muchas veces se llevan a cabo en las áreas de trabajo de los usuarios, e incluso el acceso físico no es controlado, lo que genera una falta de seguridad de los datos.

4.4 BASES DE DATOS HOMOGÉNEAS Y HETEROGÉNEAS

En los sistemas de bases de datos distribuidos homogéneas, todos los sitios emplean idéntico software de gestión de bases de datos, son conscientes de la existencia de los demás sitios y acuerdan cooperar en el procesamiento de las solicitudes de los usuarios. En estos sistemas, los sitios locales renuncian a una parte de su autonomía en cuanto a su derecho a modificar los esquemas o el software de gestión de bases de datos. Ese software también debe cooperar con los demás sitios en el intercambio de la información sobre las transacciones para hacer posible su procesamiento entre varios sitios.

A diferencia de lo anterior, en las bases de datos distribuidos heterogéneas podría ser que los diferentes sitios utilicen esquemas y software de gestión de sistemas de bases de datos diferentes. Podría ser también que algunos sitios no tengan información de la existencia del resto y que sólo proporcionen facilidades limitadas para la cooperación en el procesamiento de las transacciones. Las diferencias en los esquemas suelen constituir un problema importante para el procesamiento de las consultas, mientras que la divergencia del software supone un inconveniente para el procesamiento de transacciones que acceden a varios sitios.

En esta sección, se centrará en las bases de datos distribuidos homogéneas. Se estudiarán brevemente los aspectos del procesamiento de las consultas en los sistemas de bases de datos distribuidos heterogéneas. Los aspectos del procesamiento de las transacciones en esos sistemas se tratan más adelante.

4.5 ALMACENAMIENTO DISTRIBUIDO DE DATOS

Sea una relación **r**, que hay que almacenar en la base de datos. Existen dos enfoques del almacenamiento de esta relación en la base de datos distribuidos:

Réplica. El sistema conserva varias réplicas (copias de respaldo) idénticas de la relación y guarda cada réplica en un sitio diferente. La alternativa a las réplicas es almacenar sólo una copia de la relación.

Fragmentación. El sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

La fragmentación y la réplica pueden combinarse: las relaciones pueden dividirse en varios fragmentos y puede haber varias réplicas de cada fragmento. En los subsecciones siguientes se profundizará en cada una de estas técnicas.

4.5.1 RÉPLICA DE DATOS

Si la relación r se replica, se guarda una copia de esa relación en dos o más sitios. En el caso más extremo se tiene una réplica completa, en la que se guarda una copia en cada sitio del sistema.

Las réplicas presentan varias ventajas y desventajas.
Disponibilidad. Si alguno de los sitios que contiene la relación r falla, esa relación puede hallarse en otro sitio distinto. Por tanto, el sistema puede seguir procesando las consultas que impliquen a r, pese al fallo del sitio.

Paralelismo incrementado. En el caso en el que la mayoría de los accesos a la relación r sólo resultasen en lecturas, diferentes sitios podrían procesar en paralelo las lecturas que impliquen a r. Cuantas más réplicas de r existan, mayor será la posibili-

dad de que los datos necesarios se encuentren en el sitio en que se ejecuta la transacción. Por tanto, la réplica de los datos minimiza su transmisión entre los diferentes sitios.

Sobrecarga incrementada durante la actualización. El sistema debe asegurar que todas las réplicas de la relación r sean consistentes; en caso contrario pueden producirse cálculos erróneos. Por tanto, siempre que se actualiza r, hay que propagar la actualización a todos los sitios que contienen réplicas. El resultado es un sobrecarga incrementada. Por ejemplo, en un sistema bancario, en el que la información de las cuentas se replica en varios sitios, es necesario asegurarse de que el saldo de cada cuenta concuerde en todos ellos.

En conclusión, la réplica mejora el rendimiento de las operaciones de lectura y aumenta la disponibilidad de los datos para las transacciones de lectura. Sin embargo, las transacciones de actualización suponen una mayor sobrecarga. El control de las actualizaciones concurrentes de los datos replicados realizadas por varias transacciones resulta más complejo que en los sistemas centralizados. Se puede simplificar la gestión de las réplicas de la relación r escogiendo una de ellas como copia principal de r. Por ejemplo, en un sistema bancario, las cuentas pueden asociarse con el sitio en que se abrieron. De manera parecida, en un sistema de reserva de asientos de avión, cada vuelo puede asociarse con el sitio en que se origina. El esquema de copias principales y otras opciones del control de concurrencia distribuida.

4.5.2 FRAGMENTACIÓN DE LOS DATOS

Si la relación r se fragmenta, r se divide en varios fragmentos $r_1,\ldots, r_n$. Estos fragmentos contienen suficiente información como para permitir la reconstrucción de la relación original r. Existen dos esquemas diferentes de fragmentación de las relaciones: la fragmentación horizontal y la vertical. La fragmentación horizontal divide la relación asignando cada tupla de r a uno o más fragmentos. La fragmentación vertical divide la relación descomponiendo el esquema r de la relación r.

Estos enfoques se presentarán fragmentando la relación cuenta, con el esquema

esquema_cuenta = (numero_cuenta, nombre_sucursal, saldo)

En la fragmentación horizontal la relación r se divide en varios subconjuntos, r_1, $r_2,\ldots,r_n$. Cada tupla de la relación r debe pertenecer, como mínimo, a uno de los fragmentos, de modo que se pueda reconstruir la relación original, si fuera necesario.

A modo de ejemplo, la relación cuenta puede dividirse en varios fragmentos, cada uno de los cuales consiste en tuplas de cuentas pertenecientes a una sucursal concreta. Si el sistema bancario sólo tiene dos sucursales (Guadarrama y Cercedilla), habrá dos fragmentos diferentes:

$$\text{cuenta}_1 = \sigma_{\text{nombre\_sucursal} = \text{"Guadarrama"}} (\text{cuenta})$$

$$cuenta_2 = \sigma_{nombre\_sucursal = \text{"Cercedilla"}} (cuenta)$$

La fragmentación horizontal suele emplearse para conservar las tuplas en los sitios en que más se utilizan, para minimizar la transferencia de datos.

En conclusión, los fragmentos horizontales pueden definirse como una selección de la relación global r, es decir, se utiliza un predicado P_i para construir el fragmento

r_i:
$r_i = \sigma P_i (r)$

La relación **r** se reconstruye tornando la unión de todos los fragmentos; es decir,

$r = r_1 \cup r_2 \cup \ldots \cup r_n$

En el ejemplo, los fragmentos son disjuntos. Al cambiar los predicados de selección empleados para crear los fragmentos se puede hacer que una tupla de r dada aparezca en más de uno de los fragmentos r_i.

En su forma más sencilla, la fragmentación vertical es igual que la descomposición. La **fragmentación vertical** de r(R) implica la definición de varios subconjuntos de atributos R_1. $R_2 \ldots R_n$ del esquema R de modo que

$R = R_1 \cup R_2 \cup \ldots \cup R_n$

Cada fragmento ri de r se define mediante

$$r_i = \prod R_i(r)$$

La fragmentación debe hacerse de modo que se pueda reconstruir la relación r a partir de los fragmentos tomando la reunión natural de

$r = r_1 \bowtie r_2 \bowtie r_3 \bowtie \ldots \bowtie r_n$

Una manera de asegurar que la relación, pueda reconstruirse es incluir los atributos de la clave principal de R en cada uno de los fragmentos R_i. De manera más general, se puede utilizar cualquier superclave. Suele resultar conveniente añadir un atributo especial, denominado id_tupla, al esquema R. El valor id_tupla de cada tupla es un valor único que distingue a esa tupla de todas las demás. El atributo id_tupla, por tanto, sirve como clave candidata para el esquema aumentado y se incluye en cada uno de los fragmentos R_i. La dirección física o lógica de la tupla puede utilizarse como id_tupla, ya que cada tupla tiene una dirección única.

Para mostrar la fragmentación vertical, considérese una base de datos universitaria con una relación info_empleado que almacena, para cada empleado, id_empleado, nombre, puesto y salario. Por motivos de preservación de la confidencialidad puede que esta relación se fragmente en una denominada empleado_infoprivada, que contenga id_empleado y salario, y en otra llamada empleado_infopública, que contenga los atributos id_empleado, nombre y puesto. Puede que las dos relaciones se almacenen en sitios diferentes, nuevamente, por motivos de seguridad.

Se pueden aplicar los dos tipos de fragmentación a un mismo esquema; por ejemplo, los fragmentos obtenidos de la fragmentación horizontal de una relación pueden dividirse nuevamente de manera vertical. Los fragmentos también pueden replicarse. En general, los fragmentos pueden replicarse, las réplicas de los fragmentos pueden fragmentarse más, y así sucesivamente.

4.5.3 TRANSPARENCIA

No se debe exigir a los usuarios de los sistemas distribuidos de bases de datos, que conozcan la ubicación física de los datos ni el modo en que se puede acceder a ellos en cada sitio local concreto. Esta característica, denominada transparencia de los datos, puede adoptar varias formas:

Transparencia de la fragmentación. No se exige a los usuarios que conozcan el modo en que se ha fragmentado la relación.

Transparencia de la réplica. Los usuarios ven cada objeto de datos como lógicamente único. Puede que el sistema distribuido replique los objetos para incrementar el rendimiento del sistema. o la disponibilidad de los datos. Los usuarios no deben preocuparse por los objetos que se hayan replicado ni por la ubicación de esas réplicas.

Transparencia de la ubicación. No se exige a los usuarios que conozcan la ubicación difícil de los datos. El sistema distribuido de bases de datos debe poder hallar los datos siempre que la transacción del usuario facilite el identificador de esos datos.

Los elementos de datos (como las relaciones, los fragmentos y las réplicas) deben tener nombres únicos. Esta propiedad es fácil de asegurar en las bases de datos centralizadas. En las bases de datos distribuidos, sin embargo, hay que tener cuidado para asegurarse de que dos sitios no utilicen el mismo nombre para elementos de datos diferentes.

Una solución a este problema es exigir que todos los nombres se registren en un servidor de nombres; ubicado en el servidor central. El servidor de nombres ayuda a garantizar que el mismo nombre no se utilice para elementos de datos diferentes. También se puede utilizar el servidor de nombres para localizar los elementos de datos, dado su nombre. Este enfoque, sin embargo, presenta dos inconvenientes principales. En primer lugar, puede que el servidor de nombres se transforme en un

cuello de botella para el rendimiento cuando los elementos de datos se buscan por el nombre, lo que da lugar a un bajo rendimiento. En segundo lugar, si el servidor de nombres queda fuera de servicio, puede que ningún otro sitio del sistema distribuido logre seguir en funcionamiento.

Un enfoque alternativo más utilizado exige que cada sitio anteponga su propio identificador de sitio a cualquier nombre que genere. Este enfoque garantiza que dos sitios diferentes no generen nunca el mismo nombre (dado que cada sitio tiene un identificador único). Además, no se necesita ningún control centralizado. Esta solución, sin embargo, no logra conseguir transferencia de la ubicación, dado que a los nombres se les adjuntan los identificadores de los sitios. Así se puede hacer referencia a la relación cuenta como cuenta.sitio17, o cuenta@sitio17, en lugar de meramente cuenta. Muchos sistemas de bases de datos utilizan la dirección de Internet de los sitios para identificarlos.

Para superar este problema, el sistema de bases de datos puede crear un conjunto de nombres alternativos, o alias, para los elementos de datos. En consecuencia, los usuarios se pueden referir a los elementos de datos mediante nombres sencillos que el sistema traduce a los nombres completos. La relación entre los alias y los nombres reales puede almacenarse en cada sitio. Cuando se emplean alias, el usuario puede ignorar la ubicación física de los elementos de datos. Además, no se ve afectado si el administrador de una base de datos decide trasladar un elemento de datos de un sitio a otro (Silberschatz, Korth, Sudarshan 2006).

Los usuarios no deberían tener necesidad de hacer referencia a una réplica concreta de un elemento de datos. En vez de eso, el sistema debe determinar la réplica a la que hay que hacer referencia en las solicitudes leer, y actualizar todas las réplicas en las solicitudes escribir. Se puede asegurar que lo hace si se mantiene una tabla de catálogo, que el sistema utilizará para determinar todas las réplicas del elemento de datos. Se muestra en la figura 4.2 arquitectura del sistema siguiente:

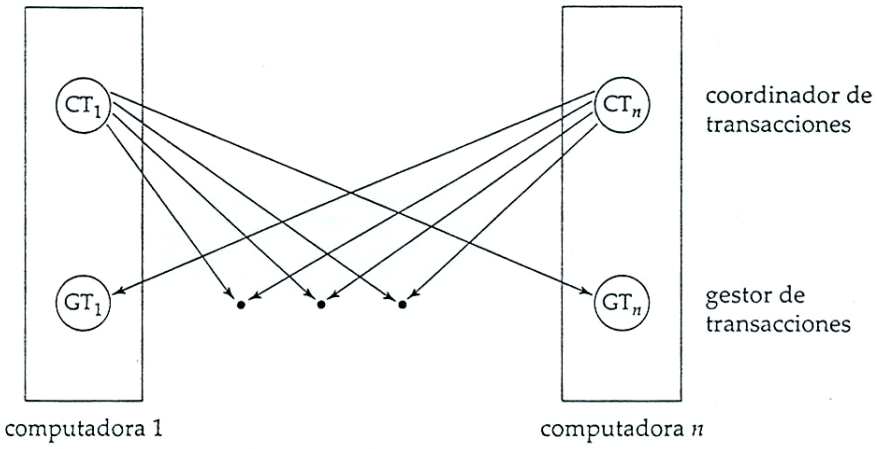


Figura 4.2. Arquitectura del sistema.

4.6 TRANSACCIONES DISTRIBUIDAS

El acceso a los diferentes elementos de datos en los sistemas distribuidos suele realizarse mediante transacciones, que deben preservar las propiedades ACID. Se deben considerar dos tipos de transacciones. Las transacciones locales son las que acceden a los datos y los actualizan en una única base de datos local; las transacciones globales son las que acceden a los datos y los actualizan en varias bases de datos locales. Las propiedades ACID de las transacciones locales pueden asegurar a la base de datos. Sin embargo, para las transacciones globales, esta tarea resulta mucho más complicada, dado que puede que participen en la ejecución de varios sitios. El fallo de alguno de estos sitios, o el de alguno de los enlaces de comunicación que los conectan entre sí, puede dar lugar a cálculos erróneos.

En esta sección, se estudia la estructura del sistema de una base de datos distribuida y sus posibles modos de fallo. Con base en el modelo presentado en este apartado, se estudia más adelante los protocolos para garantizar el compromiso atómico de las transacciones globales, y los protocolos para el control de concurrencia en las bases de datos distribuidos. Más adelante se estudia el modo en que pueden seguir funcionando las bases de datos distribuidos incluso en presencia de varios tipos de fallo.

4.6.1 ESTRUCTURA DEL SISTEMA

Cada sitio tiene su propio gestor local de transacciones, cuya función es garantizar las propiedades ACID de las transacciones que se ejecuten allí. Los diferentes gestores de transacciones colaboran para ejecutar las transacciones globales. Para comprender el modo en que se pueden implementar estos gestores, considérese un modelo abstracto de sistema de transacciones, en el que cada sitio contenga dos subsistemas:

- El gestor de transacciones gestiona la ejecución de las transacciones (o subtransacciones) que acceden a los datos almacenados en un sitio local. Se debe tener en cuenta que cada una de esas transacciones puede ser local (es decir, una transacción que se ejecuta sólo en ese sitio) o formar parte de una transacción global, es decir, una transacción que se ejecuta en varios sitios.
- El coordinador de transacciones coordina la ejecución de las diferentes transacciones (tanto locales como globales) iniciadas en ese sitio.

Ha sido demostrada en la figura 4.2 arquitectura global del sistema.

La estructura de los gestores de transacciones es parecida en muchos aspectos a la de los sistemas centralizados. Cada gestor de transacciones es responsable de:

- Mantener un registro histórico con fines de recuperación.
- Participar en un esquema adecuado de control de concurrencia para coordinar la ejecución concurrente de las transacciones que se ejecuten en ese sitio.

Como se verá, es necesario modificar tanto el esquema de recuperación como el de concurrencia para adaptar a la distribución de las transacciones.

El subsistema del coordinador de transacciones no es necesario en los entornos centralizados, ya que las transacciones sólo acceden a los datos de un sitio. Los coordinadores de transacciones, como su propio nombre implica, son responsables de la coordinación de la ejecución de todas las transacciones iniciadas en ese sitio.

En cada una de esas transacciones el coordinador es responsable de:

- Iniciar la ejecución de la transacción.
- Dividir la transacción en varias subtransacciones y distribuir esas subtransacciones a los sitios correspondientes para su ejecución.
- Coordinar la terminación de la transacción, lo que puede hacer que la transacción se comprometa o se aborte en todos los sitios.

4.6.2 MODOS DE FALLO DEL SISTEMA

Los sistemas distribuidos pueden sufrir los mismos tipos de fallos que los sistemas centralizados. Por ejemplo, errores de *software*, errores de *hardware* y fallos de discos. Sin embargo, en los entornos distribuidos también hay que tratar con otros tipos de fallos. Los tipos básicos de fallos son:

- Fallos de sitios.
- Pérdidas de mensajes.
- Fallos de enlaces de comunicaciones.
- Divisiones de la red.

En los sistemas distribuidos siempre es posible la pérdida o el deterioro de los mensajes. El sistema utiliza protocolos de control de transmisiones, como TCP/ IP, para tratar esos errores.

Sin embargo, sí dos sitios A y B no se hallan conectados de manera directa, los mensajes de uno a otro deben encaminarse mediante una serie de enlaces de comunicaciones. Si falla uno de los enlaces, hay que volver a encaminar los mensajes que debería haber transmitido. En algunos casos se puede hallar otra ruta por la red, de modo que los mensajes puedan alcanzar su destino. En otros casos, el fallo puede hacer que no haya ninguna conexión entre los dos sitios. Un sistema está dividido si se ha partido en dos (o más) subsistemas, denominados particiones, que carecen de conexión entre ellas. Obsérvese que, con esta definición, cada subsistema puede consistir en un solo nodo, etcétera.

4.7 CONTROL DE LA CONCURRENCIA EN LAS BASES DE DATOS DISTRIBUIDOS

En esta sección, se muestra el modo en que se pueden modificar algunos de los esquemas de control de concurrencia para utilizarlos en entornos distribuidos. Se da por supuesto que cada sitio participa en la ejecución de un protocolo de compromiso para garantizar la atomicidad global de las transacciones.

Los protocolos que se describen en esta sección necesitan que se hagan actualizaciones de todas las réplicas de los elementos de datos. Si ha fallado algún sitio que contenga una réplica de un elemento de datos, no se pueden procesar las actualizaciones de ese elemento de datos. Más adelante se describen los protocolos que pueden continuar el procesamiento de las transacciones aunque haya fallado algún sitio o algún enlace, lo que proporciona una gran disponibilidad.

Los diferentes protocolos de bloqueo descritos antes se pueden utilizar en entornos distribuidos. La única modificación que hay que incorporar es el modo en que el gestor de bloqueos trata los datos replicados. Se presentan varios esquemas posibles que son aplicables a entornos en que los datos se pueden replicar en varios sitios. Se dará por supuesto la existencia de los modos de bloqueo compartido y exclusivo.

4.7.1 ENFOQUE DE GESTOR ÚNICO DE BLOQUEOS

En el enfoque de gestor único de bloqueos el sistema mantiene un único gestor de bloqueos que reside en un sitio único escogido (por ejemplo, Si). Todas las solicitudes de bloqueo y de desbloqueo se realizan en el sitio Si. Cuando una transacción necesita bloquear un elemento de datos, envía una solicitud de bloqueo a Si. El gestor de bloqueos determina si se le puede conceder el bloqueo de manera inmediata. En caso afirmativo, envía un mensaje al respecto al sitio en el que se inició la solicitud de bloqueo. En caso contrario, la solicitud se retrasa hasta que se puede conceder, en cuyo momento se envía un mensaje al sitio en el que se inició la solicitud de bloqueo. La transacción puede leer el elemento de datos de cualquiera de los sitios en los que residan sus réplicas. En el caso de las operaciones de escritura, deben implicarse todos los sitios en los que residan réplicas del elemento de datos.

El esquema tiene las ventajas siguientes:
Implementación sencilla. Este esquema necesita dos mensajes para tratar las solicitudes de bloqueo y sólo uno para las de desbloqueo.

Tratamiento sencillo de los interbloqueos. Como todas las solicitudes de bloqueo y de desbloqueo se realizan en un solo sitio, se pueden aplicar directamente a este entorno los algoritmos de tratamiento de los interbloqueos.

Los inconvenientes del esquema, son:
Cuello de botella. El sitio Si, se transforma en un cuello de botella, ya que todas las solicitudes deben procesarse allí.

Vulnerabilidad. Si el sitio, Si falla, se pierde el controlador de la concurrencia. O bien hay que detener el procesamiento, o bien hay que utilizar: un esquema de recuperación para que un sitio de respaldo pueda asumir la administración de los bloqueos.

4.7.2 GESTOR DISTRIBUIDO DE BLOQUEOS

Se puede lograr un compromiso entre las ventajas y los inconvenientes ya mencionados mediante el enfoque del gestor distribuido de bloqueos, en el que la función de gestor de bloqueos se haya distribuido entre varios sitios (Silberschatz, Korth, Sudarshan 2006).

Cada sitio mantiene un gestor de bloqueos local, cuya función es administrar las solicitudes de bloqueo y de desbloqueo para los elementos de datos que se almacenan en ese sitio. Cuando, una transacción desea bloquear el elemento de datos Q, que no está replicado y reside en el sitio Si, envía un mensaje al gestor de bloqueos del sitio Si, para solicitarle un bloqueo (en un modo de bloqueo determinado). Si, el elemento de datos Q está bloqueado en un modo incompatible, la solicitud se retrasa hasta que se le pueda conceder. Una vez se ha determinado que la solicitud de bloqueo se puede conceder, el gestor de bloqueos devuelve un mensaje al sitio que ha iniciado la solicitud para indicar que le ha concedido la solicitud de bloqueo.

El esquema del gestor distribuido de bloqueos presenta la ventaja de su sencilla implementación y reduce el grado en el que el coordinador constituye un cuello de botella. Tiene una sobrecarga razonablemente baja, ya que sólo necesita dos transferencias de mensajes para tratar las solicitudes de bloqueo y una transferencia de mensaje para las de desbloqueo. Sin embargo, el tratamiento de los interbloqueos resulta más complejo, dado que las solicitudes de bloqueo y de desbloqueo ya no se realizan en un solo sitio. Puede haber interbloqueos entre sitios aunque no los haya dentro de ninguno de los sitios. Los algoritmos de tratamiento de los interbloqueos y para que detecten los interbloqueos globales.

4.7.3 COPIA PRINCIPAL

Cuando un sistema utiliza la réplica de datos se puede escoger una de las réplicas como copia principal. Así, para cada elemento de datos Q, la copia principal de Q debe residir exactamente en un sitio, que se denomina sitio principal de Q.

Cuando una transacción necesita bloquear el elemento de datos Q, solicita un bloqueo en el sitio principal de Q. Como ya se ha visto, la respuesta a la solicitud se retrasa hasta que pueda concederse.

Por tanto, la copia principal permite que el control de concurrencia de los datos replicados se trate como el de los datos no replicados. Esta semejanza permite una implementación sencilla. Sin embargo, si falla el sitio principal de Q, ese elemento de datos queda inaccesible, aunque otros sitios que contengan réplicas suyas estén accesibles.

4.8 DISPONIBILIDAD DE BASES DATOS DISTRIBUIDOS

Uno de los objetivos del empleo de bases de datos distribuidos, es disfrutar de una disponibilidad elevada; es decir, la base de datos debe funcionar casi todo el tiempo. En concreto, dado que los fallos son más probables en los sistemas distribuidos de gran tamaño, las bases de datos distribuidos deben seguir funcionando aunque sufran diferentes tipos de fallos. La posibilidad de continuar funcionando incluso durante los fallos se denomina robustez (Silberschatz, Korth, Sudarshan 2006).

Para que un sistema distribuido sea robusto debe detectar los fallos, reconfigurar el sistema de modo que el cálculo pueda continuar y recuperarse cuando se repare el procesador o el enlace.

Los diferentes tipos de fallos se tratan de manera diferente. Por ejemplo, la pérdida de mensajes se trata mediante su retransmisión. La retransmisión repetida de un mensaje por un enlace, sin la recepción de un acuse de recibo, suele ser síntoma de un fallo de ese enlace. La red intentará hallar una ruta alternativa para el mensaje. La imposibilidad de hallar esa ruta suele ser síntoma de una división de la red.

Sin embargo, no suele ser posible diferenciar claramente entre los fallos de los sitios y las divisiones de la red. El sistema, generalmente, puede detectar que se ha producido un fallo, sin embargo, puede que no logre identificar su tipo. Por ejemplo, supóngase que el sitio S1 no puede comunicar con S2. la falta de comunicación entre S1 y S2 puede ser porque que S2 ha fallado. Sin embargo, otra posibilidad es que el enlace entre S1 y S2 este fallando, lo que habrá provocado la división de la red. El problema se aborda en parte empleando varios enlaces entre los diferentes sitios, de modo que, aunque falle algún enlace, sigan conectados. Sin embargo, todavía pueden fallar varios enlaces simultáneamente, por lo que hay situaciones en las que no se puede estar seguro de que si se ha producido un fallo del sitio o una división de la red.

Sin embargo, en algunos casos, cuando los objetos de datos están replicados, puede que sea posible seguir adelante con las operaciones de lectura y de actualización aunque algunas réplicas estén inaccesibles. En ese caso, cuando se recupera el sitio que ha fallado, si tenía réplicas de algún objeto de datos, debe obtener los valores actualizados de esos objetos de datos y asegurarse de que recibe todas las actualizaciones posteriores.

- Si los datos replicados se guardan en un sitio que ha fallado o que está inaccesible, hay que actualizar el catálogo para que las consultas no hagan referencia a la copia ubicada en ese sitio. Cuando el sitio vuelva a estar activo, hay que asegurarse de que los datos que alberga sean consistentes.

- Si el sitio que ha fallado es un servidor central de algún subsistema, hay que celebrar una elección para determinar el nuevo servidor. Entre los servidores centrales están los servidores de nombres, los coordinadores de concurrencia y los detectores globales de interbloqueos.

Dado que, en general, no es posible distinguir entre los fallos de los enlaces de red y los de los sitios, se tienen que diseñar todos los esquemas de reconfiguración para que funcionen de manera correcta en caso de división de la red. En concreto, deben evitarse las situaciones siguientes:

- Que se elijan dos o más servidores centrales en particiones distintas.
- Que más de una partición actualice un mismo elemento de datos replicado.

4.9 PROCESAMIENTO DISTRIBUIDO DE CONSULTAS

Que existe una gran variedad de métodos para el cálculo de la respuesta a una consulta. Se examinaron varias técnicas para escoger una estrategia de procesamiento de consultas que minimice el tiempo que se tarda en calcular la respuesta. Para los sistemas centralizados el criterio principal para medir el costo de una estrategia dada es el número de accesos a disco. En los sistemas distribuidos hay que tener en cuenta varios asuntos más, entre los que se incluyen:

- El costo de la transmisión de los datos por la red.
- La posible ganancia de rendimiento al hacer que varios sitios procesen en paralelo diferentes partes de la consulta.

El costo relativo de la transferencia de datos por la red y del intercambio de datos con el disco varía ampliamente en función del tipo de red y de la velocidad de los discos. Por tanto, en general, uno no se puede centrar exclusivamente en los costos de disco ni tampoco en los costos de la red. Más bien hay que hallar un buen equilibrio entre los dos.

4.9.1 TRANSFORMACIÓN DE CONSULTAS

Considérese una consulta extremadamente sencilla: "hallar todas las tuplas de la relación cuenta". Aunque la consulta es sencilla (en realidad, trivial), su procesamiento no es trivial, ya que puede que la relación cuenta esté fragmentada, replicada o ambas cosas, como se ha visto antes. Si la relación cuenta está replicada, hay que elegir la réplica. Si las réplicas no se han dividido, se escoge aquélla para la que el

costo de transmisión sea mínimo. Sin embargo, si alguna réplica se ha dividido, la elección no resulta tan sencilla, ya que hay que calcular varias reuniones o uniones para reconstruir la relación cuenta. En ese caso, el número de estrategias para el sencillo ejemplo escogido puede ser elevado. Puede que la optimización de las consultas mediante la enumeración exhaustiva de todas las estrategias alternativas no resulte práctica en esas situaciones (Silberschatz, Korth, Sudarshan 2006).

La transparencia de la fragmentación implica que los usuarios pueden escribir una consulta como

nombre_sucursal = "Guadarrama" (cuenta)

Ya que cuenta está definida como
cuenta1 cuenta2

la expresión que resulta del esquema de traducción de nombres es

nombre_sucursal = "Guadarrama" (cuenta1 cuenta2)

Esta expresión, se puede simplificar de manera automática mediante las técnicas de optimización de consultas. El resultado es la expresión

nombre_sucursal = ”Guadarrama" (cuenta1) nombre_sucursal = "Guadarrama" (cuenta2)

La cual incluye dos subexpresiones. La primera sólo implica a cuenta1 y, por tanto, puede evaluarse en el sitio Guadarrama. La segunda sólo implica a cuenta2 y, por tanto, puede evaluarse en el sitio Cercedilla.

Hay otra optimización más que puede hacerse al evaluar

nombre_sucursaL = "Guadarrama" (cuenta1)

Dado que cuenta1, sólo contiene tuplas correspondientes a la sucursal de Guadarrama, se puede eliminar la operación de selección. Al evaluar

nombre_sucursal = "Guadarrama" (cuenta2)

se puede aplicar la definición del fragmento de cuenta2 para obtener

nombre_sucursal = "Guadarrama" (nombre_sucursal = "Cercedilla” (cuenta))

Esta expresión es el conjunto vacío, independientemente del contenido de la relación cuenta.

Por tanto, la estrategia final es que el sitio Guadarrama devuelva cuenta1 como resultado de la consulta.

4.10 BASES DE DATOS DISTRIBUIDOS HETEROGÉNEOS

Muchas de las últimas aplicaciones de bases de datos, necesitan datos de gran variedad de bases de datos ya existentes, ubicadas en un conjunto heterogéneo de entornos de **hardware** y de *software*. El tratamiento de la información ubicada en bases de datos distribuidos heterogéneos exige una capa de software adicional por encima de los sistemas de bases de datos ya existentes. Esta capa de software se denomina sistema de bases de datos múltiples. Puede que, los sistemas locales de bases de datos empleen modelos lógicos, lenguajes de definición para tratamiento de datos diferentes, que difieran en sus mecanismos de control de concurrencia y de administración de las transacciones. Los sistemas de bases de datos múltiples crean la visión de la integración lógica de las bases de datos sin necesidad de su integración física.

La integración completa de sistemas heterogéneos en una misma base de datos distribuidos homogéneos puede resultar difícil o imposible:

Dificultades técnicas. La inversión en los programas de aplicaciones basados en los sistemas de bases de datos ya existentes puede ser enorme, y el costo de transformar esas aplicaciones puede resultar prohibitivo.

Dificultades organizativas. Aunque la integración resulte técnicamente posible, puede que no lo sea políticamente, porque los sistemas de bases de datos ya existentes pertenezcan a diferentes empresas u organizaciones. En ese caso es importante que el sistema de bases de datos múltiples permita que los sistemas de bases de datos locales conserven un elevado grado de autonomía para la base de datos local y para las transacciones que se ejecuten con esos datos.

Por los motivos anteriores los sistemas de múltiples bases de datos ofrecen ventajas significativas que compensan la sobrecarga que suponen. En esta sección se proporciona una visión general de los retos que se afrontan al construir entornos con bases de datos múltiples desde el punto de vista de la definición de los datos y del procesamiento de las consultas. Se ofrece una visión general de los problemas de la administración de las transacciones en múltiples bases de datos.

4.10.1 VISTA UNIFICADA DE LOS DATOS

Cada sistema local de administración de bases de datos puede utilizar un modelo de datos diferente. Por ejemplo, algunos pueden emplear el modelo relacional, mientras que otros pueden emplear modelos de datos más antiguos, como el de red o el jerárquico.

Dado que, se supone que los sistemas con múltiples bases de datos ofrecen la visión de un solo sistema de bases de datos integrado, hay que utilizar un modelo de datos común. Una opción adoptada con frecuencia es el modelo relacional, que tiene como software de consulta al **Lenguaje Estructurado de Consulta** (SQL). En realidad hoy en día, hay varios sistemas disponibles que permiten realizar consultas SQL en sistemas de administración de bases de datos no relacionales.

Otra dificultad es proporcionar un esquema conceptual común. Cada sistema local ofrece su propio esquema conceptual. El sistema de bases de datos múltiples debe integrar esos esquemas independientes en uno común. La integración de los esquemas es una tarea complicada, sobre todo por la heterogeneidad semántica.

La integración de los esquemas no es la simple traducción directa de unos lenguajes de definición de datos a otros. Podría ser que los mismos nombres de atributos aparezcan en diferentes bases de datos locales con significados distintos. Puede que los tipos de datos utilizados en un sistema no estén soportados por los demás y, que la traducción de unos tipos a otros no resulta sencilla. Incluso en el caso de tipos de datos idénticos pueden surgir problemas debidos a la representación física de los datos: puede que un sistema utiliza ASCII y otro EBCDIC; las representaciones en coma flotante pueden ser diferentes, los enteros pueden representarse como de orden **más significativo** (big-endian) o como de orden **menos significativo** (little-endian). En el nivel semántico, el valor entero de una longitud puede ser pulgadas en un sistema y milímetros en otro, lo que crea una situación incómoda en la que la igualdad entre los enteros sea sólo un concepto aproximado (como ocurre siempre con los números con coma flotante). Puede que el mismo nombre aparezca en idiomas distintos en diferentes sistemas. Por ejemplo, puede que un sistema basado en Estados Unidos se refiera a la ciudad de Saragossa, mientras que uno con base en España lo haga como Zaragoza.

Todas estas diferencias aparentemente menores deben registrarse de manera adecuada en el esquema conceptual global común. Hay que proporcionar funciones de traducción. Hay que anotar los índices para el comportamiento dependiente del sistema (por ejemplo, el orden de clasificación de los caracteres no alfanuméricos no es igual en ASCII que en EBCDIC). Como ya se ha comentado, puede que la alternativa de convertir cada base de datos a un formato común no resulte factible sin dejar obsoletos los programas de aplicación ya existentes (Silberschatz, Korth, Sudarshan 2006).

4.10.2 PROCESAMIENTO DE CONSULTAS

El procesamiento de consultas en las bases de datos heterogéneos puede resultar complicado, algunos de los problemas que se pueden presentar son:

- Podría ser que algunos orígenes de datos sólo ofrezcan capacidades de consulta limitadas; por ejemplo, puede que soporten selecciones y no reuniones. Puede incluso que restrinjan dada una consulta en un esquema global, puede que haya

que traducir la consulta a consultas en los esquemas locales de cada uno de los sitios en que hay que ejecutar la consulta. Habría que volver a traducir los resultados de las consultas al esquema global.

- La tarea se simplifica escribiendo envolturas para cada origen de datos, que ofrezcan una vista de los datos locales en el esquema global. Las envolturas traducen las consultas del esquema global a consultas del esquema local y vuelven a traducir los resultados al esquema global. Las envolturas pueden ofrecerlas cada sitio o escribirse de manera independiente como parte del sistema de múltiples bases de datos.

- Las envolturas pueden, incluso, utilizarse para proporcionar una vista relacional de orígenes de datos no relacionales, como las páginas *web* (posiblemente con interfases de formularios), archivos planos, bases de datos jerárquicos o de red y sistemas de directorio.

- Podría ser que algunos orígenes de datos sólo ofrezcan capacidades de consulta limitadas; por ejemplo, puede que soporten selecciones y no reuniones. Puede incluso que restrinjan la forma de las selecciones, permitiéndolas sólo para determinados campos; los orígenes de datos web con interfases de formulario son un ejemplo de este tipo de orígenes de datos. Por tanto, puede que haya que dividir las consultas para que se lleven a cabo en parte en el origen de datos y en parte en el sitio que formula la consulta.

- En general, puede ser que ha accedido a más de un sitio para responder a una consulta dada. Es posible que haya que procesar las respuestas obtenidas de los diferentes sitios para eliminar los valores duplicados. Supóngase que un sitio contiene las tuplas de cuenta que satisfacen la selección saldo < 100, mientras que otro contiene las que satisfacen saldo > 50. Una consulta sobre toda la relación cuenta exigiría acceder a los dos sitios y eliminar las respuestas duplicadas consecuencia de las tuplas con saldo entre 50 y 100, que están replicadas en los dos sitios.

- La optimización global de consultas en bases de datos heterogéneas resulta difícil, ya que puede que el sistema de ejecución de consultas no conozca los costos de los planes de consulta alternativos en los diferentes sitios. La solución habitual es confiar sólo en la optimización en el ámbito local y utilizar únicamente la heurística en el ámbito global.

Los sistemas mediadores son sistemas que integran varios orígenes de datos heterogéneos, lo que proporciona una vista global integrada de los datos y facilidades de consulta sobre la misma. A diferencia de los sistemas de bases de datos múltiples completos, los sistemas mediadores no se ocupan del procesamiento de las transacciones (los términos mediador y bases de datos múltiples suelen utilizarse de

manera indistinta, y puede que los sistemas denominados mediadores soporten formas limitadas de transacción). El término base de datos virtual se utiliza para hacer referencia a los sistemas de bases de datos múltiples o a los sistemas mediadores, ya que ofrecen la apariencia de una sola base de datos con un esquema global, aunque los datos estén en varios sitios en esquemas locales.

4.11 SISTEMAS DE DIRECTORIO

Considérese una organización que desea poner los datos de sus empleados a disposición de diferentes miembros de la organización; entre esos datos estarían el nombre, el cargo, el ID de empleado, la dirección, la dirección de correo electrónico, el número de teléfono, el número de fax, etcétera. En los días anteriores a la informática las organizaciones creaban directorios físicos de los empleados y los distribuían por toda la organización. Incluso en nuestros días, las compañías telefónicas crean directorios físicos de sus clientes.

En general, un directorio es un listado de la información sobre alguna clase de objetos como las personas. Los directorios pueden utilizarse para hallar información sobre un objeto concreto o, en sentido contrario, hallar objetos que cumplen un determinado requisito. En el mundo de los directorios telefónicos físicos, los directorios que permiten las búsquedas en sentido directo se denominan páginas blancas, mientras que los directorios que permiten las búsquedas en sentido inverso se denominan páginas amarillas.

En el mundo interconectado de hoy en día, la necesidad de los directorios sigue vigente y, si acaso, es aún más importante. Sin embargo, hoy en día los directorios deben estar disponibles en las redes informáticas en lugar de estar en forma física (papel).

CAPÍTULO V

5. GESTIÓN DE TRANSACCIONES

5.1 INTRODUCCIÓN

La transacción hace referencia a un conjunto de operaciones que forman una única unidad lógica de trabajo. Por ejemplo, la transferencia de dinero de una cuenta a otra es una transacción que consta de dos actualizaciones, una para cada cuenta (Silberschatz, Korth, Sudarshan 2006).

Resulta importante que, o bien se ejecuten completamente todas las acciones de una transacción, o bien, en caso de fallo, se deshagan los efectos parciales de cada transacción incompleta. Esta propiedad se denomina atomicidad. Además, una vez ejecutada con éxito una transacción, sus efectos deben persistir en la base de datos, un fallo en el sistema no debe tener como consecuencia que la base de datos descarte una transacción que se ha completado con éxito. Esta propiedad se denomina durabilidad.

En los sistemas de bases de datos, en los que se ejecutan de manera concurrente varias transacciones, si no se controlan las actualizaciones de los datos compartidos, existe la posibilidad de que las transacciones operen sobre estados intermedios inconsistentes creados por las actualizaciones de otras transacciones. Esta situación puede dar lugar a actualizaciones erróneas de los datos almacenados en la base de datos. Por tanto, los sistemas de bases de datos deben proporcionar los mecanismos para aislar las transacciones de otras transacciones que se ejecuten de manera concurrente. Esta propiedad se denomina aislamiento.

Las transacciones a menudo, desde el punto de vista del usuario de una base de datos, se consideran a un conjunto de varias operaciones sobre una base de datos como una única operación. Por ejemplo, una transferencia de fondos desde una cuenta corriente a una cuenta de ahorros es una operación simple desde el punto de vista del cliente; sin embargo, en el sistema de base de datos, está compuesta internamente por varias operaciones. Evidentemente es esencial que tengan lugar todas las operaciones o que, en caso de fallo, ninguna de ellas se produzca. Sería inaceptable efectuar el cargo de la transferencia en la cuenta corriente y que no se abonase en la cuenta de ahorros.

Se ha denominado transacción, a una colección de operaciones que forman una única unidad lógica de trabajo. Un sistema de base de datos debe asegurar que la ejecución de las transacciones se realice adecuadamente a pesar de la existencia de fallos, o se ejecuta la transacción completa o no se ejecuta en absoluto. Además, debe gestionar la ejecución concurrente de las transacciones evitando introducir inconsistencias. Volviendo al ejemplo de la transferencia de fondos, una transacción que calcule el saldo total del cliente podría ver el saldo de la cuenta corriente antes

de que sea cargado por la transacción de la transferencia de fondos, y el saldo de la cuenta de ahorros después del abono. Como resultado, se obtendría un resultado incorrecto.

5.2 CONCEPTO DE TRANSACCIÓN

Se define una transacción, como una unidad de ejecución de un programa que accede y posiblemente actualiza varios elementos de datos. Una transacción se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación (por ejemplo, SQL, C++ o Java), y está delimitado por instrucciones (o llamadas a función) de la forma **inicio de transacción** (begin transaction) y **fin de transacción** (end transaction). La transacción consiste en todas las operaciones que se ejecutan entre begin transaction y end transaction (Silberschatz, Korth, Sudarshan 2006).

Para asegurar la integridad de los datos se necesita que el sistema de base de datos mantenga las siguientes propiedades de las transacciones:

a) **Atomicidad**. O bien todas las operaciones de la transacción se realizan adecuadamente en la base de datos o ninguna de ellas.

b) **Consistencia.** La ejecución aislada de la transacción (es decir, sin otra transacción que se ejecute concurrentemente) conserva la consistencia de la base de datos.

c) **Aislamiento**. Aunque se ejecuten varias transacciones concurrentemente, el sistema garantiza que para cada par de transacciones Ti y Tj, se cumple que para los efectos de Ti, o bien Tj ha terminado su ejecución antes de que comience Ti, o bien que Tj ha comenzado su ejecución después de que Ti termine. De este modo, cada transacción ignora al resto de las transacciones que se ejecuten concurrentemente en el sistema.

d) **Durabilidad**. Tras la finalización con éxito de una transacción, los cambios realizados en la base de datos permanecen, incluso si hay fallos en el sistema.

Estas propiedades, a menudo reciben el nombre de propiedades: **atomicidad, consistencia, aislamiento y durabilidad** (ACID: Atomocity, Consistency, Isolation and Durability) respectivamente.

Para comprender mejor las propiedades ACID y la necesidad de dichas propiedades, considérese un sistema bancario simplificado constituido por varias cuentas, un conjunto de transacciones que acceden y actualizan dichas cuentas. Por ahora se asume que la base de datos reside permanentemente en disco, sin embargo, una porción de la misma reside temporalmente en la memoria principal.

El acceso a la base de datos se lleva a cabo mediante las dos operaciones siguientes:

leer(X), que transfiere el dato X, de la base de datos a una memoria intermedia local perteneciente a la transacción que ejecuta la operación leer.

escribir(X), que transfiere el dato X, desde la memoria intermedia local de la transacción que ejecuta la operación escribir a la base de datos.

En un sistema de base de datos real, la operación escribir no tiene por qué producir necesariamente una actualización de los datos en disco; la operación escribir puede almacenarse temporalmente en memoria y llevarse a disco más tarde. Sin embargo, por el momento se supondrá que la operación escribir actualiza inmediatamente la base de datos.

Sea Ti, una transacción para transferir 50 $ de la cuenta A a la cuenta B. Se puede definir dicha transacción como

```
Ti: leer(A);
    A = A - 50;
    escribir(A);
    leer(B);
    B := B + 50;
    escribir(B).
```

a) **Atomicidad**. Se supone que justo antes de ejecutar la transacción Ti, los valores de las cuentas A y B son de 1,000 $ y de 2,000 $, respectivamente. Supóngase ahora que durante la ejecución de la transacción Ti, se produce un fallo que impide que dicha transacción finalice con éxito su ejecución. Ejemplos de este tipo de fallos pueden ser los fallos en el fluido eléctrico, los fallos del *hardware* y los errores *software*. Además, se supone que el fallo tiene lugar después de ejecutarse la operación escribir(A), sin embargo, antes de ejecutarse la operación escribir (B). Es ese caso, los valores de las cuentas A y B, que se ven reflejados en la base de datos son $ 950 y $ 2.000. Se ha perdido $ 50 de la cuenta A, como resultado de este fallo. En particular se puede ver que ya no se conserva la suma A + B.

Así, como resultado del fallo, el estado del sistema deja de reflejar el estado real del mundo que se supone que modela la base de datos. Un estado así se denomina estado inconsistente. Hay que asegurarse de que estas inconsistencias no sean visibles en un sistema de base de datos. Obsérvese, sin embargo, que un sistema puede en algún momento alcanzar un estado inconsistente. Incluso si la transacción Ti se ejecuta por completo, existe un punto en el que el valor de la cuenta A es de $ 950 y el de la cuenta B es de $ 2.000, lo cual constituye claramente un estado inconsistente. Este estado, sin embargo, se sustituye eventualmente por otro estado consistente en el que el valor de la cuenta A es de $ 950 y el de la cuenta B es de $ 2.050. De este modo, si la transacción no empieza nun-

ca o se garantiza que se complete, un estado inconsistente así, no será visible excepto durante la ejecución de la transacción. Ésta es la razón, de que aparezca el requisito de atomicidad. Si, se proporciona la propiedad de atomicidad, o todas las acciones de la transacción se ven reflejadas en la base de datos, o ninguna de ellas.

La idea básica para asegurar la atomicidad es la siguiente. El sistema de base de datos mantiene los valores antiguos (en disco) de aquellos datos sobre los que una transacción realiza una escritura y, si la transacción no completa su ejecución, los valores antiguos se recuperan para que parezca que la transacción no se ha ejecutado. Estas ideas se muestran más adelante. La responsabilidad de asegurar la atomicidad es del sistema de base de datos; en concreto, lo maneja un componente llamado componente de gestión de transacciones.

b) **Consistencia.** En este caso, el requisito de consistencia es que la suma de A y B no sea alterada al ejecutar la transacción. Sin el requisito de consistencia, la transacción podría crear o destruir dinero. Se puede comprobar fácilmente que si una base de datos es consistente antes de ejecutar una transacción, sigue siéndolo después de ejecutar dicha transacción.

La responsabilidad de asegurar la consistencia de una transacción es del programador de la aplicación que codifica dicha transacción. La comprobación automática de las restricciones de integridad puede facilitar esta tarea.

c) **Aislamiento**. Incluso si se aseguran las propiedades de consistencia y de atomicidad para cada transacción, si varias transacciones se ejecutan concurrentemente, se pueden entrelazar sus operaciones de un modo no deseado, produciendo un estado inconsistente.

Por ejemplo, como se ha visto antes, la base de datos es inconsistente temporalmente durante la ejecución de la transacción para transferir fondos de la cuenta A a la cuenta B, con el total deducido escrito ya en A y el total incrementado todavía sin escribir en B. Si una segunda transacción que se ejecuta concurrente leen A y B en este punto intermedio y calcula A + B, observará un valor inconsistente. Además, si esta segunda transacción realiza después modificaciones en A y B basándose en los valores leídos, la base de datos puede permanecer en un estado inconsistente aunque ambas transacciones terminen.

Una solución para el problema de ejecutar transacciones concurrentemente es ejecutarlas secuencialmente, es decir, una tras otra. Sin embargo, la ejecución concurrente de transacciones produce notables beneficios en el rendimiento, como se verá más adelante. Se han desarrollado otras soluciones que permiten la ejecución concurrente de varias transacciones.

Los problemas que causa la ejecución concurrente de transacciones. La propiedad de aislamiento asegura que el resultado obtenido al ejecutar concurrente-

mente las transacciones es un estado del sistema equivalente a uno obtenido al ejecutar tras otra en algún orden. Los principios de aislamiento se presentarán más adelante. La responsabilidad de asegurar la propiedad de aislamiento es de un componente del sistema de base de datos llamado componente de control de concurrencia.

d) **Durabilidad.** Una vez que se completa con éxito la ejecución de una transacción, y después de comunicar al usuario que inició la transacción que se ha realizado la transferencia de fondos, un fallo en el sistema no debe producir la pérdida de datos correspondientes a dicha transferencia.

La propiedad de durabilidad asegura que, una vez que se completa con éxito una transacción, persisten todas las modificaciones realizadas en la base de datos, incluso si hay un fallo en el sistema después de completarse la ejecución de dicha transacción.

A partir de ahora se asume que un fallo en el sistema produce una pérdida de datos de la memoria principal, sin embargo, los datos almacenados en disco nunca se pierden. Se puede garantizar la durabilidad si se asegura que:

- Las modificaciones realizadas por la transacción se guardan en disco antes de que finalice la transacción.

- La información de las modificaciones realizadas por la transacción guardada en disco es suficiente para permitir a la base de datos reconstruir dichas modificaciones cuando el sistema se reinicie después del fallo.

La responsabilidad de asegurar la durabilidad pertenece a un componente software del sistema de base de datos llamado gestor de recuperaciones. El componente de gestión de transacciones y el componente de gestión de recuperaciones están estrechamente relacionados, así, como su implementación.

5.3 ESTADOS DE UNA TRANSACCIÓN

En ausencia de fallos, todas las transacciones se completan con éxito. Sin embargo, como se ha visto antes, una transacción puede que no siempre termine su ejecución con éxito. Una transacción de este tipo se denomina abortada. Si se pretende asegurar la propiedad de atomicidad, una transacción abortada, no debe tener efecto sobre el estado de la base de datos. Así, cualquier cambio que ha hecho la transacción abortada sobre la base de datos debe deshacerse. Una vez que se han deshecho los cambios efectuados por la transacción abortada, se dice que la transacción está retrocedida. Parte, de la responsabilidad del esquema de recuperaciones es gestionar las transacciones abortadas (Silberschatz, Korth, Sudarshan 2006).

Una transacción que termina con éxito se dice que está comprometida. Una transacción comprometida que ha hecho modificaciones transforma la base de datos llevándola a un nuevo estado consistente, que permanece incluso si hay un fallo en el sistema.

Cuando una transacción se ha comprometido no se pueden deshacer sus efectos abortándola. La única forma de deshacer los cambios de una transacción comprometida es ejecutando una transacción compensadora. Por ejemplo, si una transacción añade $ 20 a una cuenta, la transacción compensadora debería restar $ 20 de la cuenta. Sin embargo, no siempre se puede crear dicha transacción compensadora. Por tanto, se deja al usuario la responsabilidad de crear y ejecutar transacciones compensadoras.

Es necesario precisar qué se entiende por terminación con éxito de una transacción. Se establece, por tanto, un simple modelo abstracto de transacción. Una transacción debe estar en uno de los estados siguientes:
Activa. El estado inicial; la transacción permanece en este estado durante su ejecución.
Parcialmente comprometida. Después de ejecutarse la última instrucción.
Fallida. Tras descubrir que no puede continuar la ejecución normal.
Abortada. Después del retroceso de la transacción y de haber restablecido la base de datos a su estado anterior al comienzo de la transacción.
Comprometida. Tras completarse con éxito.

El diagrama de estados correspondiente a una transacción, se muestra en la figura 5.1, se dice que una transacción se ha comprometido solo si ha llegado al estado de comprometida. Análogamente se dice que una transacción ha abortado sólo si ha llegado al estado de abortada. Una transacción se dice que ha terminado si se ha comprometido o bien se ha abortado.

Una transacción comienza en el estado activa. Cuando acaba su última instrucción pasa al estado de parcialmente comprometida. En este punto la transacción ha terminado su ejecución, sin embargo, es posible que aún tenga que ser abortada, puesto que los datos actuales pueden estar todavía en la memoria principal y puede producirse un fallo en el *hardware* antes de que se complete con éxito.

El sistema de base de datos escribe en disco la información suficiente para que, incluso al producirse un fallo, puedan reproducirse los cambios hechos por la transacción al reiniciar el sistema tras el fallo. Cuando se termina de escribir esta información, la transacción pasa al estado de comprometida (Álvarez 2006).

Como se ha mencionado antes, se asume que los fallos no provocan pérdidas de datos en disco. Las técnicas para tratar las pérdidas de datos en disco.

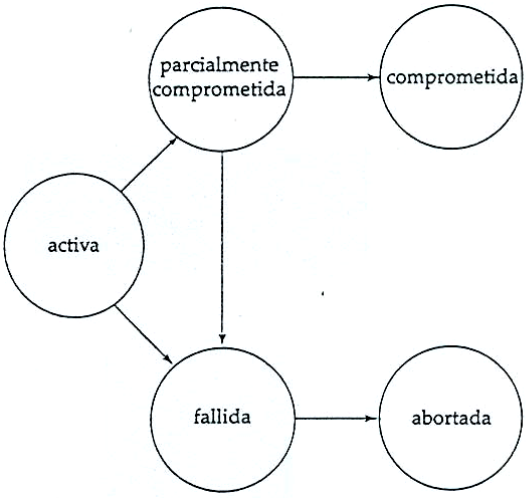


Figura 5.1. Diagrama de transición de estado de una transacción.

Una transacción llega al estado fallida después de que el sistema determine que dicha transacción no puede continuar su ejecución normal, por ejemplo, a causa de errores de *hardware* o lógicos. Una transacción de este tipo se debe retroceder. Después pasa al estado abortado. En este punto, el sistema tiene dos opciones:

- Reiniciar la transacción, sin embargo, sólo si la transacción se ha abortado a causa de algún error *hardware o software* que no lo ha provocado la lógica interna de la transacción. Una transacción reiniciada se considera una nueva transacción.

- Cancelar la transacción. Normalmente se hace esto si hay algún error interno lógico que sólo se puede corregir escribiendo de nuevo el programa de aplicación, o debido a una entrada incorrecta o a que no se han encontrado los datos deseados en la base de datos.

Hay que tener cuidado cuando se trabaja con escrituras externas observables, como en un terminal o en una impresora. Cuando una escritura así tiene lugar, no puede borrarse puesto que puede haber sido vista fuera del sistema de base de datos. Muchos sistemas permiten que tales escrituras tengan lugar sólo después de que la transacción llegue al estado de comprometida. Una manera de implementar dicho esquema es hacer que el sistema de base de datos almacene temporalmente cualquier valor asociado con estas escrituras externas en memoria no volátil, y realiza las escrituras actuales sólo si la transacción llega al estado de comprometida. Si el sistema falla después de que la transacción llegue al estado comprometida, sin embargo, antes de que finalicen las escrituras externas, el sistema de base de datos puede llevar a cabo dichas escrituras externas (utilizando los datos de la memoria no volátil) cuando el sistema se reinicie.

La gestión de las escrituras externas puede ser más complicada en ciertas situaciones. Por ejemplo, supóngase que la acción externa es dispensar dinero en un cajero automático y el sistema falla justo antes de que se dispense el dinero (se asume que el dinero se dispensa atómicamente). No tiene sentido dispensar el dinero cuando se reinicie el sistema, ya que el usuario probablemente no esté en el cajero. En tal caso es necesaria una transacción compensadora, como devolver el dinero a la cuenta del usuario.

Para algunas aplicaciones puede ser deseable permitir a las transacciones activas que muestren datos a los usuarios, particularmente para transacciones de larga duración que se ejecutan durante minutos u horas. Pero, no se puede permitir dicha salida de datos observables a no ser que se quiera arriesgar la atomicidad de la transacción. Muchos sistemas de bases de datos actuales aseguran la atomicidad y, por tanto, prohíben esta forma de interacción con el usuario.

5.4 IMPLEMENTACIÓN DE LA ATOMICIDAD Y LA DURABILIDAD

El componente de gestión de recuperaciones de un sistema de base de datos implementa el soporte para la atomicidad y durabilidad. En primer lugar se considera un esquema simple, sin embargo, es extremadamente ineficiente, denominado copia respaldo (sombra). Este esquema, que se basa en hacer copias de la base de datos, denominadas copias respaldos, asume que sólo una transacción está activa en cada momento. El esquema asume que la base de datos es simplemente un archivo en disco. En disco se mantiene un puntero llamado **puntero_bd** que apunta a la copia actual de la base de datos.

En el esquema de copia respaldo, una transacción que quiera actualizar una base de datos crea primero una copia completa de dicha base de datos. Todos los cambios se hacen en la nueva copia de la base de datos dejando la copia original, la copia respaldo inalterado. Si, en cualquier punto hay que abortar la transacción, la copia nueva simplemente se borra. La copia antigua de la base de datos no se ve afectada.

Si la transacción se completa, se compromete como sigue. En primer lugar se consulta al sistema operativo para asegurarse de que todas las páginas de la nueva copia de la base de datos se han escrito en disco (en los sistemas Unix se utiliza el comando fsync para este propósito). Después de terminar este comando se actualiza el puntero_bd para que apunte a la nueva copia de la base de datos; la nueva copia se convierte entonces en la copia de la base de datos actual. La copia antigua de la base de datos se borra después. El esquema se describe en la figura 5.2, en la cual se muestra el estado de la base de datos antes y después de la actualización. Se dice que la transacción está comprometida en el momento en que puntero_bd actualizado se escribe en disco.

Considérese ahora la manera en que esta técnica trata los fallos de las transacciones y del sistema. En primer lugar, considérese un fallo en la transacción. Si la transacción falla en algún momento antes de actualizar el puntero_bd, el contenido de la base de datos anterior no se ve afectado. Se puede abortar la transacción simplemente borrando la nueva copia de la base de datos. Una vez que se ha comprometido la transacción, puntero_bd apunta a todas las modificaciones que ésta ha realizado en la base de datos. De este modo, o todas las modificaciones de la transacción se ven reflejadas o ninguna dé ellas o independientemente del fallo de la transacción.

Considérese, ahora el resultado de un fallo en el sistema. Supóngase, que el sistema falla en algún momento antes de escribir en disco el puntero_bd actualizado. Entonces, cuando se reinicie el sistema; leerá el puntero_bd y verá el contenido original de la base de datos, y ninguno de los efectos de la transacción será visible en la base de datos. A continuación supóngase, que el sistema falla después de actualizar en disco el puntero_bd. Antes, de que el puntero se actualice, todas las páginas actualizadas de la nueva copia de la base de datos se escriben en disco. De nuevo, se asume que una vez que un archivo se escribe en disco, su contenido no se daña incluso si hay un fallo del sistema. Por tanto, cuando el sistema se reinicie, leerá el puntero_bd y verá el contenido de la base de datos después de la transacción que ha realizado todas las modificaciones (Silberschatz, Korth, Sudarshan 2006).

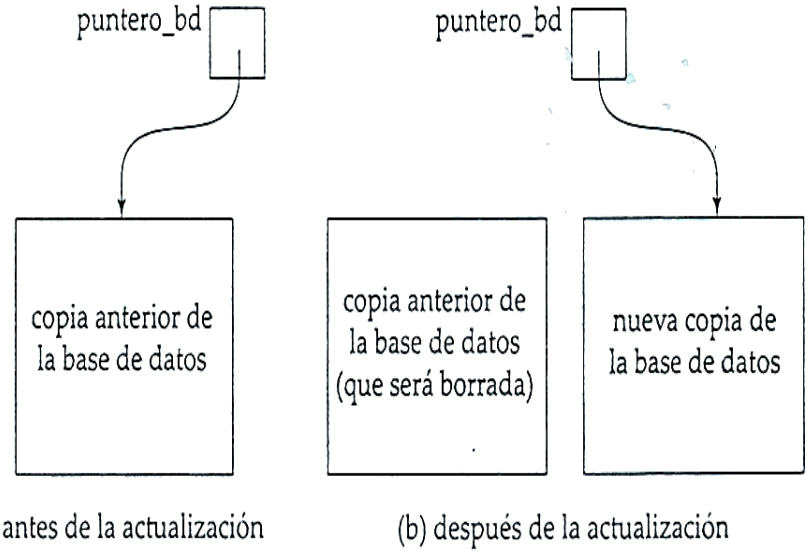


Figura 5.2. Técnica de copia en la sombra para la atomicidad y durabilidad.

La implementación de la copia respaldo depende realmente de que escribir el puntero_bd sea una operación atómica; es decir, o se escriben todos sus octetos o ninguno de ellos. Si se escribieran algunos de los octetos del puntero y otros no, el puntero no tendría sentido y al reiniciarse el sistema no se podrían encontrar ni la versión anterior ni la nueva de la base de datos. Afortunadamente los sistemas de disco proporcionan actualizaciones atómicas de bloques enteros, o al menos de un

sector del disco. En otras palabras, el sistema de disco garantiza la actualización automática del puntero_bd, siempre, que se cumpla que el puntero_bd queda en un único sector, lo que se puede asegurar almacenándolo al comienzo de un bloque.

De este modo, la implementación de la copia respaldo (sombra) del componente de gestión de recuperaciones, asegura las propiedades de atomicidad y durabilidad de las transacciones. Como ejemplo simple, fuera del dominio de las bases de datos, considérese, una sesión de edición de texto. Una sesión completa de edición de texto puede modelar una transacción. Las acciones que ejecuta la transacción son leer y actualizar el archivo. Guardar el archivo cuando se termina de editar significa completar la transacción de edición; terminar la sesión de edición sin guardar el archivo significa abortar la transacción de edición.

Muchos editores de texto, utilizan fundamentalmente la implementación que se acaba de describir para asegurar que una sesión de adición es como una transacción. Se utiliza un nuevo archivo para almacenar las modificaciones. Al finalizar la sesión de edición, si el archivo modificado se va a guardar, se utiliza un comando para renombrar el nuevo archivo para que tenga el nombre del archivo actual. Se asume, que el renombramiento está implementado como una operación atómica en el sistema de archivos, y que al mismo tiempo borra el archivo antiguo.

Sin embargo, esta implementación es extremadamente ineficiente en el contexto de grandes bases de datos, ya que la ejecución de una simple transacción requiere copiar la base de datos completa. Además, la implementación no permite a las transacciones ejecutarse concurrentemente unas con otras. Existen formas prácticas de implementar la atomicidad y durabilidad que son mucho menos costosas y más potentes.

5.5 EJECUCIONES CONCURRENTES

Los sistemas de procesamiento de transacciones permiten normalmente la ejecución de varias transacciones concurrentemente, es decir permiten que varias transacciones se actualizan concurrentemente, los datos provoca complicaciones en la consistencia de los mismos, como se ha visto antes. Asegurar la consistencia a pesar de la ejecución concurrente de las transacciones requiere un trabajo extra; es mucho más sencillo exigir que las transacciones se ejecuten secuencialmente, es decir, una a una, comenzado cada una sólo después de que la anterior se ha completado (Silberschatz, Korth, Sudarshan 2006). Sin embargo, existen dos buenas razones para permitir la concurrencia:

Productividad y utilización de recursos mejorados. Una transacción consiste en varios pasos. Algunos implican operaciones de E/S; otros implican operaciones de CPU. La CPU y los discos pueden trabajar en paralelo en una computadora. Por tanto, las operaciones de E/S se pueden realizar en paralelo con el procesamiento de la CPU. Se puede entonces explotar el paralelismo de la CPU y del sistema de E/S pa-

ra ejecutar varias transacciones en paralelo. Mientras una transacción ejecuta una lectura o una escritura en un disco, otra puede ejecutarse en la CPU mientras una tercera transacción ejecuta una lectura o una escritura en otro disco. Todo esto incrementa la productividad (rendimiento) del sistema, es decir, en el número de transacciones que puede ejecutar en un tiempo dado Análogamente, la utilización del procesador y del disco aumenta también; en otras palabras, el procesador y el disco está menos tiempos desocupados o sin hacer ningún trabajo útil.

Tiempo de espera reducida. Por lo general, en un sistema se ejecutan una serie de transacciones, algunas cortas y otras largas. Si las transacciones se ejecutan secuencialmente, la transacción corta debe esperar a que la transacción larga anterior se complete, lo cual puede llevar a un retardo impredecible en la ejecución de la transacción. Si las transacciones operan en partes diferentes de la base de datos es mejor hacer que se ejecuten concurrentemente, compartiendo los ciclos de la CPU y los accesos a disco entre ambas. La ejecución concurrente reduce los retardos impredecibles en la ejecución de las transacciones. Además, se reduce el tiempo medio de respuesta: el tiempo medio desde que una transacción comienza hasta que se completa.

La razón para usar la ejecución concurrente en una base de datos es esencialmente la misma que para usar multiprogramación en un sistema operativo.

Cuando se ejecutan varias transacciones concurrentemente, la consistencia de la base de datos se puede destruir a pesar de que cada transacción individual sea correcta. En esta sección se presenta el concepto de planificación que ayudan a identificar aquellas ejecuciones que garantizan que se asegura la consistencia.

El sistema de base de datos debe controlar la interacción entre las transacciones concurrentes para evitar que se destruya la consistencia de la base de datos. Esto se lleva a cabo a través de una serie de mecanismos denominados esquemas de control de concurrencia.

Se considera nuevamente el sistema bancario simplificado, el cual tiene varias cuentas y un conjunto de transacciones que acceden y modifican dichas cuentas. Sean T1 y T2, dos transacciones para transferir fondos de una cuenta a otra. La transacción T1 transfiere $ 50 de la cuenta A a la cuenta B y se define como sigue:

```
T1: leer(A);
    A := A - 50;
    escribir(A);
    leer(B);
    B:= B + 50;
    escribir(B).
```

La transacción T2 , transfiere el diez por ciento del saldo de la cuenta A a la cuenta B, y se define

```
T2: leer(A);
    temp:= A*0,1;
    A:= A – temp;
    escribir(A);
    leer(B);
    B:= B + temp;
    escribir(B).
```

Supóngase, que los valores actuales de las cuentas A y B son $ 1.000 y $ 2.000 respectivamente.

Supóngase, que las dos transacciones se ejecutan de una en una en el orden T1 seguida de T2. Esta secuencia de ejecución se representa en la figura 5.3. En esta figura la secuencia de pasos o instrucciones aparece en orden cronológico de arriba abajo, con las instrucciones de T1 en la columna izquierda y las de T2 en la derecha. Los valores finales de las cuentas A y B, después de que tenga lugar la ejecución de la figura 5.3, son de $ 855 y de $ 2.145 respectivamente. De este modo, la suma total de saldo de las cuentas A y B, es decir, la suma A + B se conserva tras la ejecución de ambas transacciones.

T_1	T_2
leer(A)	
$A := A$ - 50	
escribir(A)	
leer(B)	
$B := B$ + 50	
escribir(B)	
	leer(A)
	$temp := A * 0,1$
	$A := A$ - $temp$
	escribir(A)
	leer(B)
	$B := B + temp$
	escribir(B)

Figura 5.3. Planificación 1, una planificación en la que T2 sigue a T1.

Análogamente, si las transacciones se ejecutan de una en una en el orden T2 seguida de T1, entonces la secuencia de ejecución es en la figura 5.4: De nuevo, como se esperaba, se conserva la suma A + B y los valores finales de las cuentas A y B son de $ 850 y de $ 2.150 respectivamente.

T_1	T_2
	leer(A)
	$temp := A * 0,1$
	$A := A$ - $temp$
	escribir(A)
	leer(B)
	$B := B + temp$
	escribir(B)
leer(A)	
$A := A$ - 50	
escribir(A)	
leer(B)	
$B := B$ + 50	
escribir(B)	

Figura 5.4. Planificación 2, una planificación secuencial en la cual T1 sigue a T2.

Las secuencias de ejecución que se acaban de describir se denominan planificaciones. Representan el orden cronológico en el que se ejecutan las instrucciones en el sistema. Obviamente una planificación para un conjunto de transacciones debe consistir en todas las instrucciones de dichas transacciones, y debe conservar el orden en que aparecen las instrucciones en cada transacción individual. Por ejemplo en la transacción T1, la instrucción escribir(A) debe aparecer antes de la instrucción leer(B), en cualquier planificación válida. En los párrafos siguientes, planificación se referirá a la primera secuencia de ejecución (T1 seguida de T2) y planificación 2 a la segunda secuencia de ejecución (T2 seguida de T1).

Estas planificaciones son secuenciales. Cada planificación secuencial consiste en una secuencia de instrucciones de varias transacciones, en la cual las instrucciones pertenecientes a una única transacción están juntas en dicha planificación. De este modo, para un conjunto de n transacciones existen n planificaciones secuenciales válidas distintas.

Cuando el sistema de bases de datos ejecuta concurrentemente varias transacciones, la planificación correspondiente, no tiene porque ser secuencial. Si dos transacciones se ejecutan concurrentemente, el sistema operativo puede ejecutar una transacción durante un tiempo pequeño, luego realizar un cambio de contexto, ejecutar la segunda transacción durante un tiempo, cambiar de nuevo a la primera transacción durante un tiempo y así sucesivamente. Si hay muchas transacciones, todas ellas comparten el tiempo de la CPU.

Son posibles muchas secuencias de ejecución, puesto que varias instrucciones de ambas transacciones se pueden intercalar. En general no es posible predecir exactamente cuántas instrucciones se ejecutarán antes de que la CPU cambie a otra

transacción. Así, el número de planificaciones posibles para un conjunto de n transacciones es mucho mayor que n.

Volviendo, al ejemplo anterior supóngase, que las dos transacciones se ejecutan concurrentemente. Una posible planificación se muestra en la figura 5.5. Una vez que la ejecución tiene lugar, se llega al mismo estado que cuando las transacciones se ejecutan secuencialmente en el orden T1, seguido de T2. La suma A + B se conserva igualmente.

No todas las ejecuciones concurrentes producen un estado correcto. Como ejemplo de esto considérese, la planificación de la figura 5.6. Después de ejecutarse esta planificación, se llega a un estado cuyos valores finales de las cuentas A y B, son de $ 950 y de $ 2.100 respectivamente. Este estado final es un estado inconsistente, ya que se ha ganado $ 50 al procesar la ejecución concurrente. Realmente la ejecución de las dos transacciones no conserva la suma A + B.

T_1	T_2
leer(A)	
$A := A - 50$	
escribir(A)	
	leer(A)
	$temp := A * 0.1$
	$A := A - temp$
	escribir(A)
leer(B)	
$B := B + 50$	
escribir(B)	
	leer(B)
	$B := B + temp$
	escribir(B)

Figura 5.5. Planificación 3, una planificación concurrente equivalente a la planificación 1.

Si se deja el control de la ejecución concurrente completamente al sistema operativo, son posibles muchas planificaciones, incluyendo las que dejan a la base de datos en un estado inconsistente, como la que se acaba de describir. Es una tarea del sistema de base de datos asegurar que cualquier planificación que se ejecute lleve a la base de da datos a un estado consistente. El componente del sistema de base de datos que realiza esta tarea se denomina componente de control de concurrencia.

Se puede asegurar la consistencia de la base de datos en una ejecución concurrente, si, se está seguro de que cualquier planificación que se ejecute tiene el mismo efecto que otra que se hubiese ejecutado sin concurrencia. Es decir, la planificación debe ser, en cierto modo, equivalente a una planificación secuencial.

5.6 COMPROBACIÓN DE LA SECUENCIALIDAD

Cuando se diseñan esquemas de control de concurrencia, es necesario que demostrar que las planificaciones que genera el esquema son secuenciables. Para lograrlo se debe entender primero la forma de determinar si, dada una planificación concreta P, es secuenciable.

A continuación, se presenta un método simple y eficiente de determinar la secuencialidad en cuanto a conflictos de una planificación. Considérese una planificación P. Se construye un grafo dirigido, llamado grafo de precedencia para P. Este grafo consiste en un par G = (V, A), siendo V un conjunto de vértices y A un conjunto de arcos. El conjunto de vértices consiste en todas las transacciones que participan en la planificación. El conjunto de arcos consiste en todos los arcos Ti→Tj para los cuales se dan una de las tres condiciones siguientes:

1. Ti ejecuta escribir(Q) antes de que Tj ejecute leer(Q).
2. Ti ejecuta leer(Q) antes de que Tj ejecute escribir(Q).
1. Ti ejecuta escribir(Q) antes de que Tj ejecute escribir(Q).

Si existe un arco Ti→Tj en el grafo de precedencia, entonces en toda planificación secuencial P' equivalente a P, Ti debe aparecer antes de Tj.

Por ejemplo, en la figura 5.6 (a) se muestra el grafo de precedencia de la planificación 1. Sólo contiene el arco T1→T2, puesto que todas las instrucciones de T1 se ejecutan antes de que lo haga la primera de T2. Análogamente, la figura 5.6 (b) que muestra el grafo de precedencia de la planificación 2 con el único arco T2→T1, ya que todas las instrucciones de T2 se ejecutan antes de que lo haga la primera de T1.

Figura 5.6. Grafo de precedencia para (a) la planificación 1 y (b) la planificación 2.

El grafo de precedencia de la planificación 4 se representa en la figura 5.7. Contiene el arco T1→T2 debido a que T1 ejecuta leer(A) antes de que T2 ejecute escribir(A). También contiene el arco T2→T1 debido a que T2 ejecuta leer(B) antes de que T1 ejecute escribir(B).

Si el grafo de precedencia de P tiene un ciclo, entonces la planificación P no es secuenciable en cuanto a conflictos. Si el grafo no contiene ciclos, entonces la planificación P es secuenciable en cuanto a conflictos.

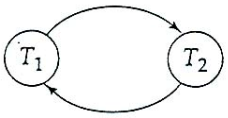
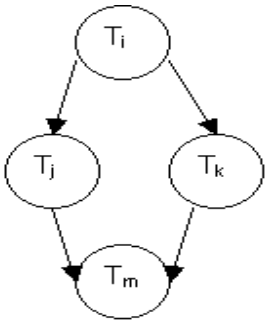
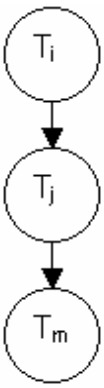


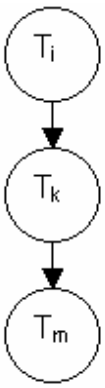
Figura 5.7. Grafo de precedencia para la planificación 4.

El orden de secuencialidad, se puede obtener a través de la ordenación topológica, la cual determina un orden lineal que consiste en el orden parcial del grafo de precedencia. En general, se pueden obtener muchos órdenes lineales posibles a través de la ordenación topológica. Por ejemplo, el grafo de la figura 5.8 (a) tiene dos órdenes lineales aceptables, como se observa en las figuras 5.8 (b) y 5.8 (c).



(a)





(b) (c)

Figura 5.8. Muestra la ordenación topológica.

Así, para probar la secuencialidad en cuanto a conflictos es necesario construir el grafo de precedencia e invocar a un algoritmo de detección de ciclos. Los algoritmos de detección de ciclos, tales como los que se basan en la búsqueda primero en profundidad, requieren del orden de n_2 operaciones, siendo n el número de vértices del

grafo (es decir, el número de transacciones). De este modo se consigue un esquema práctico para determinar la secuencialidad en cuanto a conflictos.

Volviendo a los ejemplos anteriores, obsérvese, que los grafos de precedencia para las planificaciones 1 y 2 (figura 5.6) no contienen ciclos. El grafo de precedencia para la planificación 4 (figura 5.7), sin embargo, contiene ciclos, lo que indica que esta planificación no es secuenciable en cuanto a conflictos.

La comprobación de la secuencialidad en cuanto a vistas es más complicada. De hecho, se ha demostrado que el problema de determinar la secuencialidad en cuanto a vistas es NP completo. Por tanto, seguramente no exista ningún algoritmo eficiente para comprobar la secuencialidad en cuanto a vistas.

En conclusión, una transacción es una unidad de ejecución de un programa que accede y posiblemente actualiza varios elementos de datos. Es fundamental comprender el concepto de transacción para entender e implementar las actualizaciones de los datos en una base de datos, de manera que las ejecuciones concurrentes y los fallos de varios tipos no den como resultado que la base de datos se vuelva inconsistente (Silberschatz, Korth, Sudarshan 2006).

Puesto que una transacción es una unidad que conserva la consistencia, una ejecución secuencial de transacciones garantiza que se conserve dicha consistencia.

El componente de gestión de control de concurrencia de la base de datos es el responsable de manejar los esquemas de control de concurrencia.

CAPÍTULO VI

6. METODOLOGÍA PARA IMPLEMENTAR SIMULADOR PROTOTIPO DE APLICACIÓN UTILIZANDO ALGORITMOS GENÉTICOS PARA OPTIMIZAR PROCESO TRANSACCIÓN EN LA WEB

6.1 INTRODUCCIÓN

Se ha tomado como base el capítulo V y conocimientos sobre los Algoritmos Genéticos, para implementar un simulador prototipo de aplicación utilizando algoritmos genéticos para optimizar el proceso de transacción en la web.

El objetivo es implementar un simulador de transacciones orientando a la toma de decisiones del administrador de transacciones con la aplicación de algoritmos genéticos. Se usará los algoritmos genéticos para determinar que transacción se debe tomar para asignarlo en la cola de procesos.

Se asumen, ciertas restricciones que el simulador tomará como dadas. Por ejemplo, cada transacción tiene un número constante de recursos que solicitan. Cada recurso tiene una cola que administra y solo se pueden hacer 2 tipos de requerimiento: leer y escribir.

La estructura de un cromosoma consta de un grupo de alelos y cada uno corresponde con un recurso solicitado. El administrador de transacciones tomará el requerimiento por el recurso para ponerlo en cola, el que tenga un máximo de cromosoma igual a 1, es decir, cuando encuentre entre el grupo de transacciones la transacción que tenga sus alelos en 1. Se tomará como cromosoma una cadena binaria que será convertida a números enteros.

Se ha realizado un análisis de los modelos de transacciones que operan actualmente y se ha extraído tales mecanismos para llevarlo a un proceso de toma de decisiones en función de los algoritmos genéticos.

El presente documento compila todo el análisis realizado tal como los antecedentes, las restricciones que se llevan al modelo, los requerimientos y los diagramas del modelo.

6.2 SUPUESTOS Y RESTRICCIONES

a) SUPUESTOS

Las transacciones son generadas por el usuario:

- **Recepcionado**. Indica que el coordinador de transacciones ha tomado la solicitud, sin embargo, aún sus procesos no están en cola (en su totalidad).
- **En cola**. Indica que el administrador de transacciones lo esta atendiendo.
- **Atendido**. Indica que la transacción se ha ejecutado.
- **No atendido**. Indica que no se esta procesando

- La cola de cada transacción es establecida por el usuario.
- La probabilidad de mutación es establecido por el usuario.
- La probabilidad de cruzamiento es establecido por el usuario.
- El número de iteraciones son establecidos por el usuario.
- El número de alelos son establecidos por el usuario.
- Existe un administrador (despachador), el cual asigna los procesos de manera secuencial a cada nodo, conforme tiene cola disponible para almacenar los procesos en espera de ser atendidos.
- La generación de un proceso se basa en un valor aleatorio, que está conformado por un número entero cuyo rango máximo es la combinación máxima de la longitud del cromosoma.
- El despachador determina en función de los algoritmos genéticos, cual de los nodos va atender un proceso.

b) RESTRICCIONES

- Una transacción está conformada por un conjunto de procesos.
- Cada proceso de una transacción solicita un recurso.
- Los recursos son fijos
- El requerimiento por el recurso consta de 2 estados: **read**(R) y **write**(W). Un proceso ejecuta cualquiera de las 2 opciones.
- El sistema construye de forma aleatoria el requerimiento de (R) y (W) por cada proceso.
- Un **read**(R) se asume entrada en cola de manera directa.
- Un **write**(W) queda en espera de entrar en la cola hasta que el administrador de transacciones tome la decisión en función de los algoritmos genéticos, buscando la transacción tenga un cromosoma con alelos 1.
- Cada transacción tiene un tiempo máximo de ejecución, pasado el tiempo, el coordinador de transacciones ejecuta un ROLLBACK.
- El tiempo máximo de uso del recurso por el proceso se genera de manera aleatoria y es como máximo el tiempo que tiene asignado la transacción.

6.3 CARACTERÍSTICAS Y REQUERIMIENTOS DEL SISTEMA

En esta sección, se hace un breve resumen de características y requerimientos del sistema como:

- Existen un conjunto de transacciones que compiten
- Las transacciones pueden escribir: **write**(W), **read**(R)
- Una transacción puede constar de un conjunto de procesos y cada una de ellas pueden hacer escritura o lectura para un conjunto de recursos.
- El modelo debe cumplir con las propiedades que certifican una transacción valida: atomicidad, consistencia, aislamiento y durabilidad.

- Existe un conjunto de procesos de distintas transacciones que esperan por un recurso. Una transacción puede solicitar varios recursos. La idea de una transacción involucra la solicitud de varios recursos.
- Una transacción queda en espera cuando no ha obtenido todos los recursos.
- Cada recurso mantiene una cola de atención. Se definen los recursos como páginas por procesar.
- Solo un proceso a la vez puede usar el recurso. Mientras este asignado, nadie puede tomar hasta que sea liberado por el proceso.
- El sistema puede liberar un recurso si tiene mucho tiempo en espera. Si eso ocurre, la transacción aborta.
- Se usa un modelo pesimista esperando obtener todos los bloqueos.
- El coordinador de transacciones ejecuta la administración de las transacciones y coordina que transacción debe realizarse. Ejecuta la toma de decisiones usando algoritmos genéticos para decidir que transacción es seleccionada para acceder a la cola del recurso. Asimismo, que la transacción finalmente se ejecutará y tiene la posesión de todos los recursos.
- La configuración de una transacción involucra el tiempo que solicita un recurso. Si una transacción tiene la siguiente configuración: solicita recursos ABCDEF, a su vez se indica el tiempo que solicita por cada recurso. Cada recurso tiene una cola, cuando un proceso toma el recurso, no involucra que se ejecutará, el proceso debe esperar por los demás recursos. El tiempo máximo que un proceso espera es el tiempo que requiere la transacción para ejecutarse. Si excede este tiempo, el proceso se descarta y la transacción se aborta. El coordinador de transacciones ejecuta un ROLLBACK.
- Una transacción descartada va a un repositorio de transacciones descartadas.
- Una transacción exitosa involucra que ha tomado todos los recursos y ha ejecutado la operación con el tiempo máximo permitido dado por el tiempo de la transacción.
- Si un recurso acepta un requerimiento y lo pone en la cola del recurso, el cromosoma de la transacción cambia el alelo a 1, en caso contrario se mantiene en 0.
- Los estados de una transacción pueden ser:
  Rechazada. No existe cola para procesamiento, las colas están saturadas.
  No ejecutado. Aun el coordinador de transacciones no ha tomado acción sobre la solicitud.

  a) **Requerimientos del sistema**, son:
  Sistema operativo: Windows XP, Windows Vista, Windows Server 2003, Windows Server 2008.

  b) **Requerimientos para simulador prototipo**, son:
  Memoria: 512 Mb (o superior) RAM.
  Disco: 2 Mb.

  c) **Herramientas de desarrollo** son:

Compilador: Codegear 2009 Builder C++ de Borland.
Lenguaje: C++.
Diagramador de procesos: Visio 2007.

6.4 ANÁLISIS Y DISEÑO DEL SISTEMA SIMULADOR

En esta sección, se hace un breve resumen de análisis y diseño del sistema simulador como:

1. **Variables**.
   - Tiempo de respuesta de actualización, lectura, adición.
   - **Modelo de cola**. La transacción pasa a una cola.
   - Modelo de procesamiento a disco de la transacción.
   - Arquitectura del sistema de base de datos: paralelo, multihilo, multiproceso, distribuido.

2. **Arquitectura en paralelo**.
   - Memoria compartida.
   - Disco compartido.
   - Sin compartimiento.
   - **Jerárquico**. Combinación de memoria y disco.

3. **Protocolo de concurrencia**. Pesimista y optimista

En un entorno multiusuario existen dos modelos para actualizar datos en una base de datos: concurrencia optimista y concurrencia pesimista. El objeto DataSet está diseñado para fomentar el uso de la concurrencia optimista en actividades cuya ejecución tiene una larga duración, como cuando se trabaja con interacción remota y cuando los usuarios interactúan con datos.

La concurrencia pesimista implica bloquear filas en el origen de datos para impedir que otros usuarios modifiquen los datos de tal forma que el usuario actual resulte afectado. En un modelo pesimista, cuando un usuario realiza una acción que hace que se aplique un bloqueo, otros usuarios no pueden realizar acciones que entrarían en conflicto con el bloqueo hasta que el propietario del bloqueo lo libere. Este modelo se utiliza principalmente en aquellos entornos en los que hay mucha contención de datos, de manera que el costo de proteger los datos con bloqueos es menor que el costo de deshacer transacciones si se producen conflictos de concurrencia.

Por tanto, en un modelo de concurrencia pesimista, un usuario que actualiza una fila establece un bloqueo. Hasta que el usuario no haya terminado la actualización y liberado el bloqueo, nadie más podrá modificar dicha fila. Por este motivo, la concurrencia pesimista resulta más adecuada cuando los tiempos de bloqueo son cortos, como ocurre en el procesamiento de registros mediante programación. La concu-

rrencia pesimista no es una opción escalable cuando los usuarios interactúan con los datos y hacen que los registros queden bloqueados durante períodos de tiempo relativamente largos.

Por el contrario, los usuarios que utilizan la concurrencia optimista no bloquean una fila cuando la leen. Cuando un usuario desea actualizar una fila, la aplicación debe determinar si otro usuario la ha modificado o no desde que se leyó. La concurrencia optimista suele utilizarse en entornos con poca contención de datos. Esto mejora el rendimiento porque no es necesario bloquear registros, a la vez que el bloqueo de registros requiere recursos adicionales del servidor. Además, para mantener bloqueos de registros es necesaria una conexión persistente con el servidor de base de datos. Como éste no es el caso en un modelo de concurrencia optimista, las conexiones con el servidor pueden atender a un mayor número de clientes en menos tiempo.

En un modelo de concurrencia optimista, se considera que ha habido una infracción si, después de que un usuario recibe un valor de la base de datos, otro usuario modifica el valor antes de que el primer usuario haya intentado modificarlo. En el ejemplo siguiente se describe cómo el servidor resuelve una violación de concurrencia.

6.4.1 PROBAR SI HAY INFRACCIONES DE LA CONCURRENCIA OPTIMISTA

Existen varias técnicas para probar si se ha producido una infracción de la concurrencia optimista. Una de ellas consiste en incluir una columna de marca de tiempo en la tabla. Las bases de datos suelen ofrecer funcionalidad de marca de tiempo que puede utilizarse para identificar la fecha y la hora en que se actualizó el registro por última vez. Mediante esta técnica se incluye una columna de marca de tiempo en la definición de la tabla. Siempre que se actualiza el registro, se actualiza la marca de tiempo de manera que queden reflejadas la fecha y la hora actuales. Al hacer una prueba para ver si hay infracciones de la concurrencia optimista, la columna de marca de tiempo se devuelve con cualquier consulta del contenido de la tabla. Cuando se intenta realizar una actualización, se compara el valor de marca de tiempo de la base de datos con el valor de marca de tiempo original contenido en la fila modificada. Si coinciden, se realiza la actualización y se actualiza la columna de marca de tiempo con la hora actual con el fin de reflejar la actualización. Si no coinciden, se ha producido una infracción de la concurrencia optimista.

Otra técnica para probar si hay alguna infracción relacionada con la concurrencia optimista consiste en comprobar que todos los valores de las columnas originales de una fila sigan coincidiendo con los existentes en la base de datos.

1. Problemas de arquitecturas distribuidos, las máquinas están en lugares físicamente distinto.
2. Pasos para actualizar: COMMIT / ROLLBACK, existen atomicidad, consistencia, aislamiento y durabilidad.

3. Existe recuperación de fallas, cuando no se completa la transacción o falla existe el proceso de recuperación de fallas.
4. Gestor de control de concurrencia, se ocupa de controlar la ejecución de las transacciones para asegurar la consistencia de la base de datos.
5. Puntos que debe controlar el gestor de transacciones: confiabilidad, disponibilidad, tiempo de respuesta, atomicidad, permanencia.
6. El gestor de transacciones guarda un log de transacciones para proceso de recuperación en caso de fallos. O puede rehacer una transacción llamado recuperable.
7. El archivo log es una pila donde se guardan los registros y se eliminan conforme se procesan.
8. Los archivos log guardan un **Checkpoint** para hacer punto de recuperación ante una falla sin tener que recorrer todo el archivo log.
9. Los archivos log tienen las estructuras siguientes:
    - Identificador de la transacción.
    - Hora de modificación.
    - Identificador del registro afectado.
    - Tipo de acción
    - Valor anterior del registro.
    - Nuevo valor del registro.
    - Información adicional
10. Se pueden manejar 2 archivos log: uno con la imagen antes de la grabación y otro después de la grabación.
11. Una transacción se llama unidad de ejecución.
12. Las tablas tienen seguridad con atributos de lectura, modificación, eliminación, acceso a índices.
13. **Técnicas de control de concurrencia**: el optimista y el pesimista (bloqueo y marcas de tiempo).
14. La técnica por bloqueo puede bloquear por: tabla y por registro.
15. Hay dos tipos de **bloqueo: exclusivo** (para actualizar datos) o **compartido** (que no se modifique mientras se lee).
16. El algoritmo se llama bloqueo en 2 fases. La idea es bloquear todos los elementos que necesitan accesos. Si no se logran todos, se espera.
17. Los sistemas de BD pueden bloquear: un campo, un registro, una tabla o la BD en su totalidad.
18. **Protocolo de bloqueo de 2 fases**: Existen fase de crecimiento (obtener los bloqueos) y fase de decrecimiento (liberar los bloqueos).
19. Otra técnica se llama marcado de tiempo: todas las transacciones se marcan y evita un **Deadlock** y se ejecutan de manera secuencial.
20. La técnica optimista tiene 3 fases: **lectura** (realizan copia de los objetos), **validación** (si los objetos no tienen conflicto con otra transacción) y **grabación** (graba los objetos modificados).
21. Granularidad es el detalle con que se aplica la técnica de control de concurrencia.

22. Niveles de aislamiento de las transacciones: **serializables** (todas las transacciones son secuenciales, una tras otra), **lectura repetida** (existe la posibilidad de una lectura fantasma cuando después de la primera lectura se detecta que existe otra transacción que ha modificado los datos, los cuales vuelven a leerse. Esto implica que no hay seriabilidad porque no son secuenciales, aca se presenta el caso que no existe bloqueo de todos los elementos involucrados. La idea es que la segunda lectura sea igual a la primera), **lectura comprometida** (solo se lee los registros sobre los cuales puede ejecutar bloqueo) y **lectura no comprometida** (lee los registros de otra transacción que aun no está comprometida).
23. Cuando existe la posibilidad que otra transacción modifique los datos durante el proceso de lectura, la segunda transacción de detiene hasta que la primera complete la transacción. Con ello se mantiene la seriabilidad. Existe la posibilidad de lectura no repetida, es decir, cuando al volverse a leer los datos no son iguales, en este caso se ha dado un **Rollback** a la transacción.
24. SNAPSHOT ISOLATION es una alternativa al modelo SERIALIZABLE. Se extrae un SNAPSHOT sobre el cual se ejecuta la transacción. Si otra transacción lee alguno de los registros, la transacción se descarta.
25. El estado de una transacción: activa, parcialmente activa, fallida, abortada y comprometida.
26. Arquitectura de un sistema de base de datos: red, paralelo, distribuido.
27. Multitarea con múltiples procesos en una sola máquina. Esto implica multiusuario en una sola máquina.
28. Para el simulador, los bloqueos son por páginas y no por tablas o registros de tablas.
29. Componente de almacenamiento:
    - **Concurrencia**. Control de concurrencia MULTIVERSION, SNAPSHOT READ, etcétera.
    - Soporte para transacciones de ACID.
    - **Integridad referencial**. Integridad referencial por clave foránea
    - **Almacenamiento físico**. Tamaño de las páginas para las tablas e índices.
    - **Soporte de indexación**. Por ejemplo B-TREE.
    - **Caché de memoria**. Almacenamiento en caché.
30. **Granularidad de bloqueo**. Por tabla, página o registro.
31. La norma SQL-92 define niveles de consistencia que no siempre garantizan secuencialidad de:
    a. Secuenciable.
    b. Lectura repetible: solo se pueden leer registros comprometidos. Entre dos lecturas del mismo recurso, ninguna otra transacción puede modificar dicho recurso
    c. Compromiso de lectura: solo se puede leer registros comprometidos, sin embargo, no se asegura la lectura repetible
    d. Sin compromiso de lectura: se permite leer registros no comprometidos

6.4.2 TÉCNICAS DE TRANSACCIONES

i) ÁREA DE TRABAJO PRIVADA

Consiste en realizar copias de los bloques que serán utilizados dentro de una transacción de manera que se trabaje con estas copias para realizar todas las modificaciones necesarias. Todo el espacio de trabajo con la información que será utilizada esta contenido dentro de estas copias denominado área de trabajo privada. Los demás usuarios trabajaran con la copia original de los bloques, sin embargo, no podrán obtener una segunda copia de los mismos.

Al iniciarse la transacción el proceso obtiene una copia privada de los datos. Lecturas y escrituras sobre la zona privada. Para optimizar solo se crean copias privadas de los datos modificados. Una segunda optimización para los datos que están organizados en bloques apuntados desde un índice, como los archivos, es crear copias solo de los fragmentos modificados.

6.4.3 BITÁCORA DE ESCRITURA ANTICIPADA

Este método consiste en realizar una copia con todas las transacciones que van siendo ejecutadas hacia un bloque o espacio (log) de trabajo que sea estable, esta lista se la conoce como lista de intenciones.

Las transacciones serán actualizadas con la información una vez que se ha determinado el fin de la transacción.

Se modifican los datos, sin embargo, antes se escribe en un log sobre memoria estable la descripción de la operación. En el log se escriben registros para indicar el inicio y fin de la transacción, cuando se aborta la transacción se recorre el log para deshacer los cambios. Después de una caída temporal, se debe recorrer el log. Si una transacción no ha escrito su registro de fin se aborte, si lo ha escrito, se hacen los cambios pendientes. Para evitar recorrer todo el log después de un fallo temporal de la máquina, se usan generalmente ***checkpoint.***

6.4.4 PROTOCOLO DE COMPROMISO DE DOS FASES

En un sistema distribuido, una transacción puede afectar a varios procesadores lo cual dificulta la atomicidad. La solución más típica es el protocolo de **compromiso de dos fases** (C2F). En este protocolo existe un coordinador que normalmente es el proceso que inicio la transacción.

Fase 1. El coordinador escribe en el log almacenado en memoria estable el registro (preparar T). Manda un mensaje con ese contenido a los nodos implicados en la transacción. Cada proceso implicado decide si esta listo para hacer el compromiso,

escribe en su log la decisión (listo T o no listo T) y le manda en un mensaje al coordinador.

Fase 2. Si el coordinador recibe alguna respuesta negativa u obtiene alguna falla de respuesta decide abortar la transacción. En caso contrario decide realizar el compromiso.

El coordinador escribe en el log la decisión y manda un mensaje a los procesos implicados. Cada proceso que recibe el mensaje escribe en su log la decisión del coordinador y realiza la acción correspondiente.

La terminación de una transacción se hace mediante la regla del compromiso global. El coordinador aborta una transacción si y solo si al menos un proceso implicado decide abortar. El coordinador hace un compromiso de la transacción si y solo si todos los participantes deciden realizar el compromiso.

Comportamiento ante un fallo de un nodo. El nodo N se recupera después de una caída transitoria y detecta que la transacción T estaba a medias. Si el log contiene (compromiso T) se realiza la transacción. Si contiene (aborta T) se aborta la transacción, si contiene (listo T) debe consultar al coordinador para determinar si se compromete o aborta la transacción, si no hay mensajes en el log se aborta.

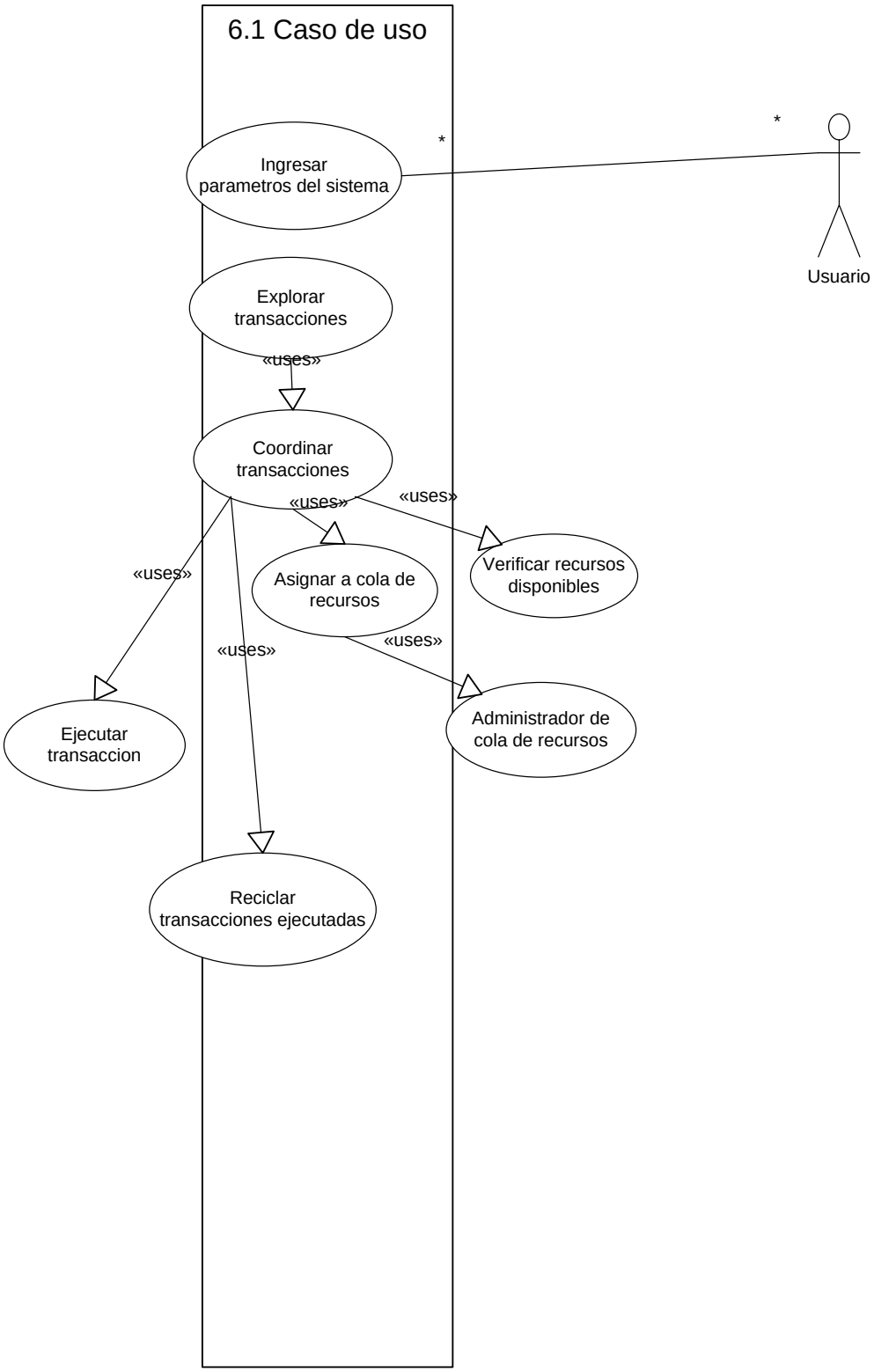
Comportamiento ante fallos del coordinador. Cada nodo implicado N debe decidir sobre la transacción T. Si el log contiene (compromiso T) se realiza la transacción. Si contiene (aborta T) se aborta, si no contiene (listo T) el coordinador no ha podido decidir el compromiso, por lo tanto, lo mas apropiado es abortar la transacción. Si todos los nodos tienen (listo T), sin embargo, ninguno tiene (compromiso T) o (aborta T) no se puede determinar la decisión del coordinador. Se debería esperar que se recuperase.

6.4.5 CARACTERIZACIÓN DEL SISTEMA SIMULADOR

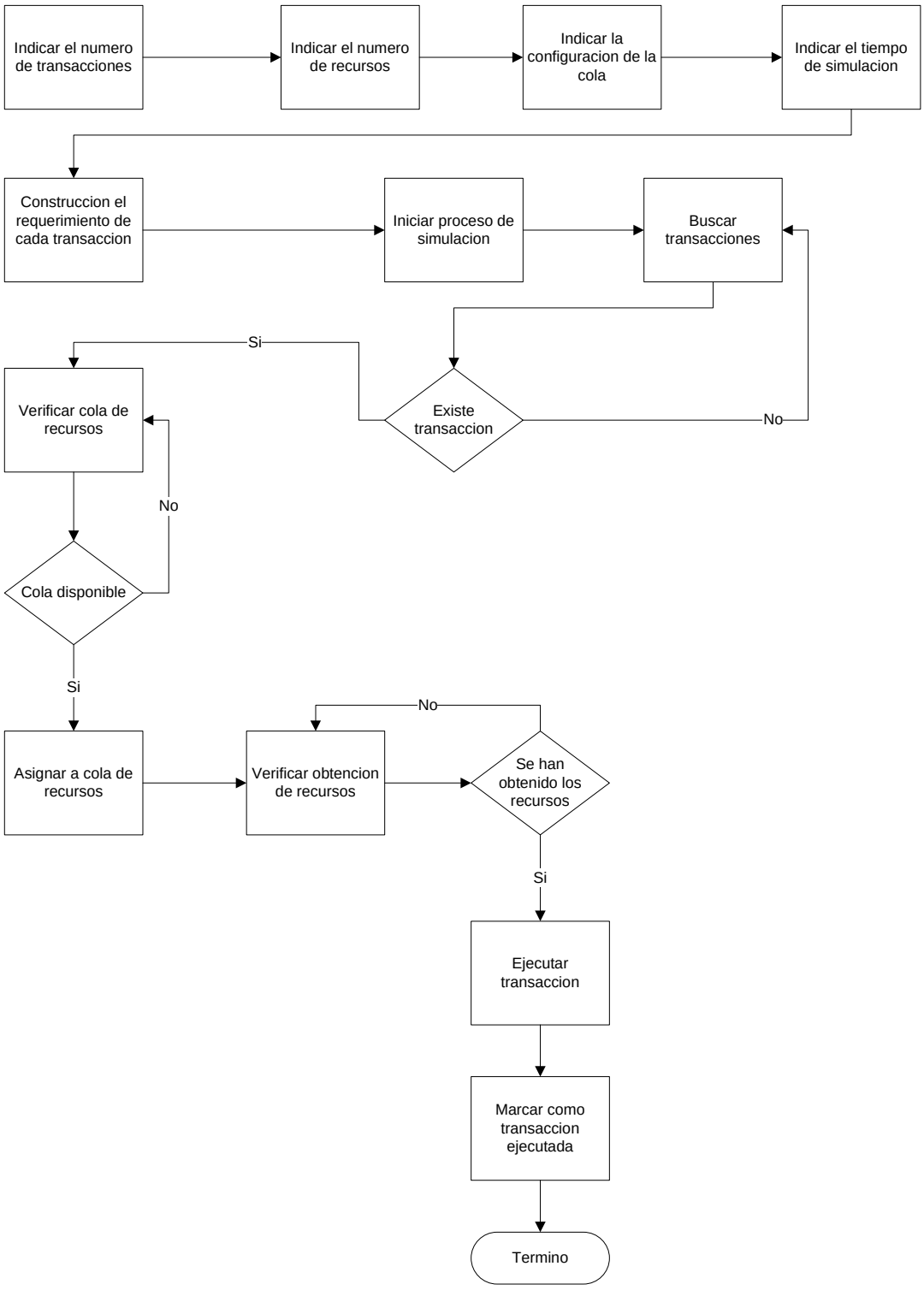
El proceso de simular la asignación de procesos de transacciones, está basado en la aplicación de algoritmos genéticos. Existe un proceso que se encarga de despachar recursos necesarios, los cuales se encargarán de agruparlos en la cola y seleccionar para procesar transacción.

El proceso de selección sobre la base de los cromosomas, aplica los algoritmos genéticos para hacer la decisión óptima de selección. El objetivo siempre es buscar el valor mayor, en una cadena binaria se convierte a un entero. El valor entero es el valor significativo para la toma de decisiones.

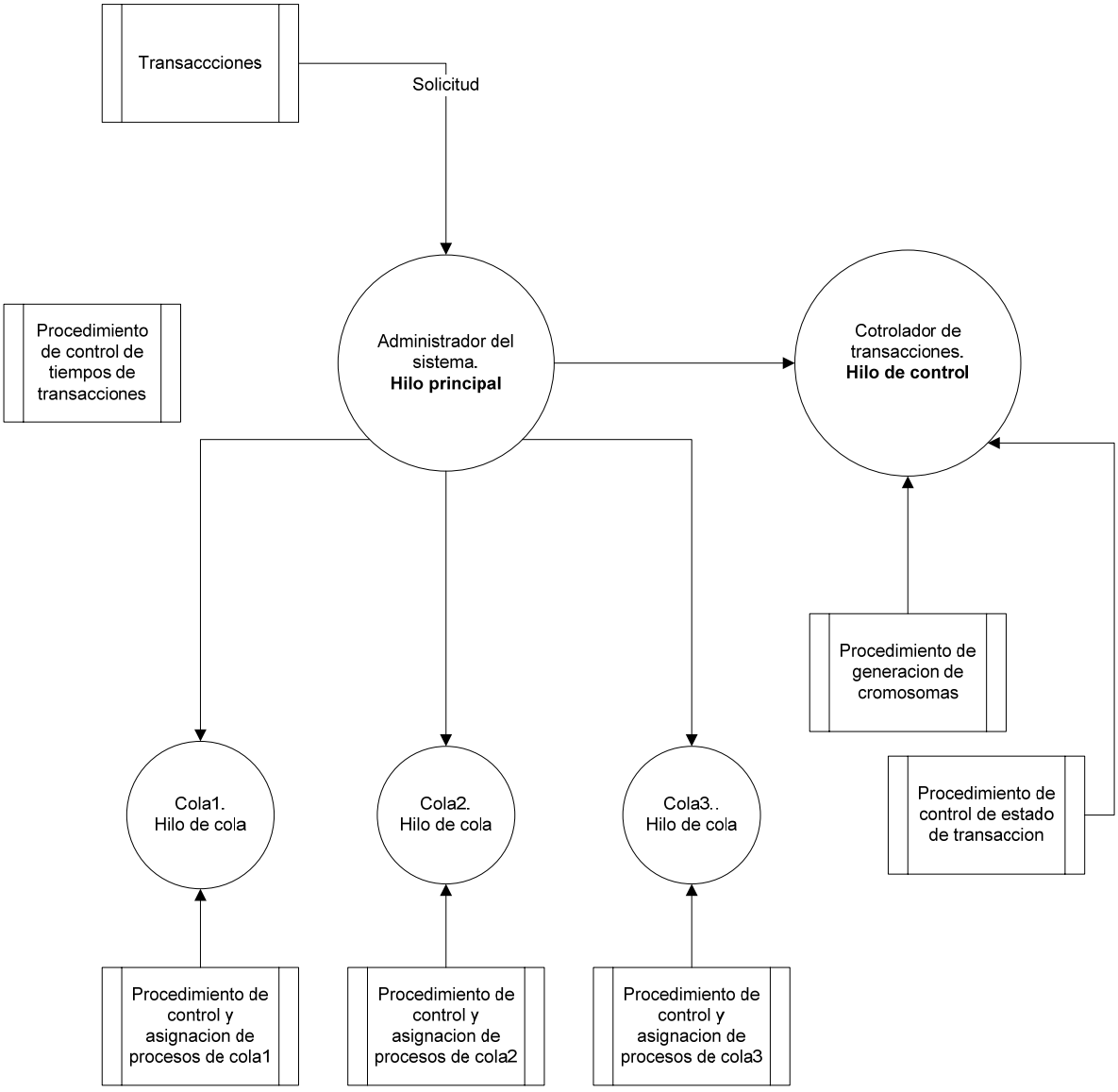
Los diagramas del simulador se muestran en la figura 6.1 caso de uso, figura 6.2 flujograma del simulador de transacciones, figura 6.3 modelo gráfico del simulador, siguientes:


6.1 Caso de uso
Ingresar parametros del sistema
Explorar transacciones
«uses»
Coordinar transacciones
«uses»
«uses»
«uses»
Asignar a cola de recursos
Verificar recursos disponibles
«uses»
«uses»
Ejecutar transaccion
Administrador de cola de recursos
Reciclar transacciones ejecutadas
*
*
Usuario

6.2 FLUJOGRAMA DEL SIMULADOR DE TRANSACCIONES



6.3 Modelo Gráfico del Simulador



6.4.5 PSEUDOCÓDIGO

```
Sistema{
Solicitar variables de inicio()
Inicializar parámetros()
Ejecutar simulador()
}

Ejecutar simulador{
Inicializar hilo principal del simulador()
Inicializar controlador de transacciones()
Inicializar hilos de las colas()
Cerrar hilo principal del simulador()
Cerrar hilo de administrador de transacciones()
Cerrar hilo de las colas()
}

Hilo principal del simulador() {
Mientras no exceda el tiempo de simulacion{
Explorar transacciones entrantes()
Existe transacciones entrantes{
                    Asignar transacción a colas de recursos()
                    Las colas están llenas y no se puede asignar la transaccion{
                        Descartar las transacciones()
}
En caso contrario{
    Recepcionar la transaccion
}
}
}
}

Hilo de las colas() {
    Explorar la cola revisando los requerimientos()
    Selecciono proceso que usar recurso()
    Excede tiempo de uso del requerimiento{
            Liberar recurso()
}
Reordenar cola con los requerimientos en espera ()
}

Hilo administrador de transacciones(){
Verificar el estado de la transacción()
Transaccion completa{
    Cambiar estado de la transacción()
```

```
Informar que la transacción esta completa()
}
}
```

CAPÍTULO VII

7. ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

7.1 PRESENTACIÓN

El sistema consta de un conjunto de pasos para llegar a ejecutarlo, está organizado de manera secuencial. Primero, se debe asignar la información para la opción Cargar configuración que esta incluido en la configuración inicial del sistema. En la opción de Ejecución puede iniciar el proceso de simulación y ver el reporte de salida en la opción Reportes. Se muestra en la figura 7.1 ventana principal del simulador, título denominado **Simulador de Transacciones** que contiene las pestañas de: Configurar, Ejecutar, Reporte, Acerca del sistema. Más abajo está la ventana de los Parámetros del sistema, y otra ventana para la Lista de transacciones y otros.

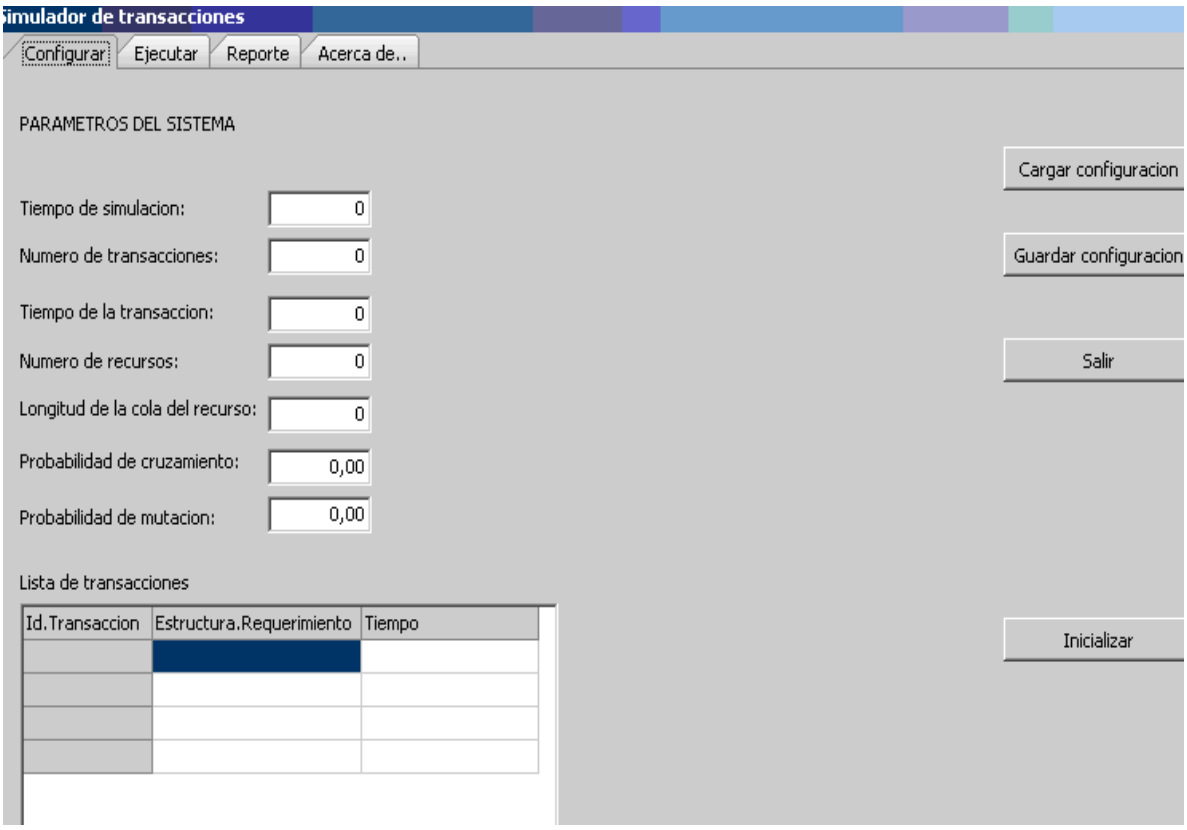


Figura 7.1. Ventana principal del simulador.

7.2 CARGAR CONFIGURACIÓN DEL SISTEMA

Este es el **paso inicial,** en la ventana principal las opciones están inicializadas con ceros cada una.

Hacer clic en **Cargar configuración** y aparecerá la ventana Open. Seleccionar la opción **testtransacciones.sim** y se muestra en la figura 7.2 ventana Open, y hacer clic sobre él, y se muestra en la figura 7.3 la ventana de parámetros del sistema siguiente:

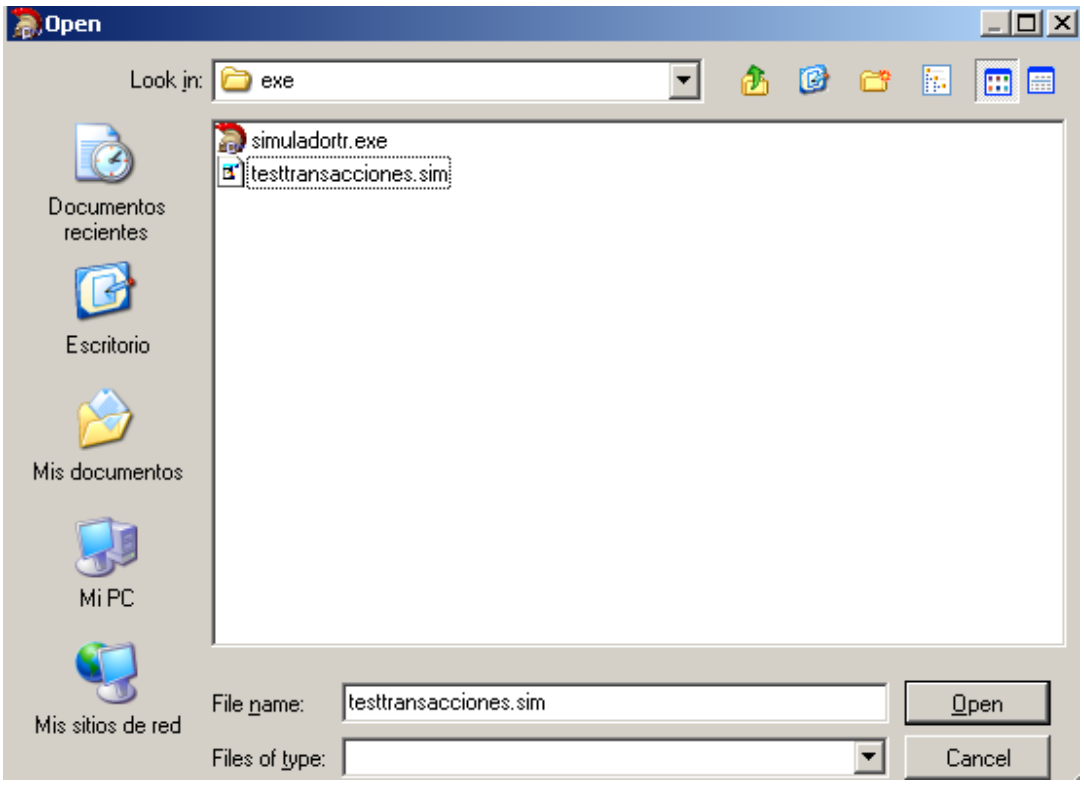


Figura 7.2. Ventana Open.

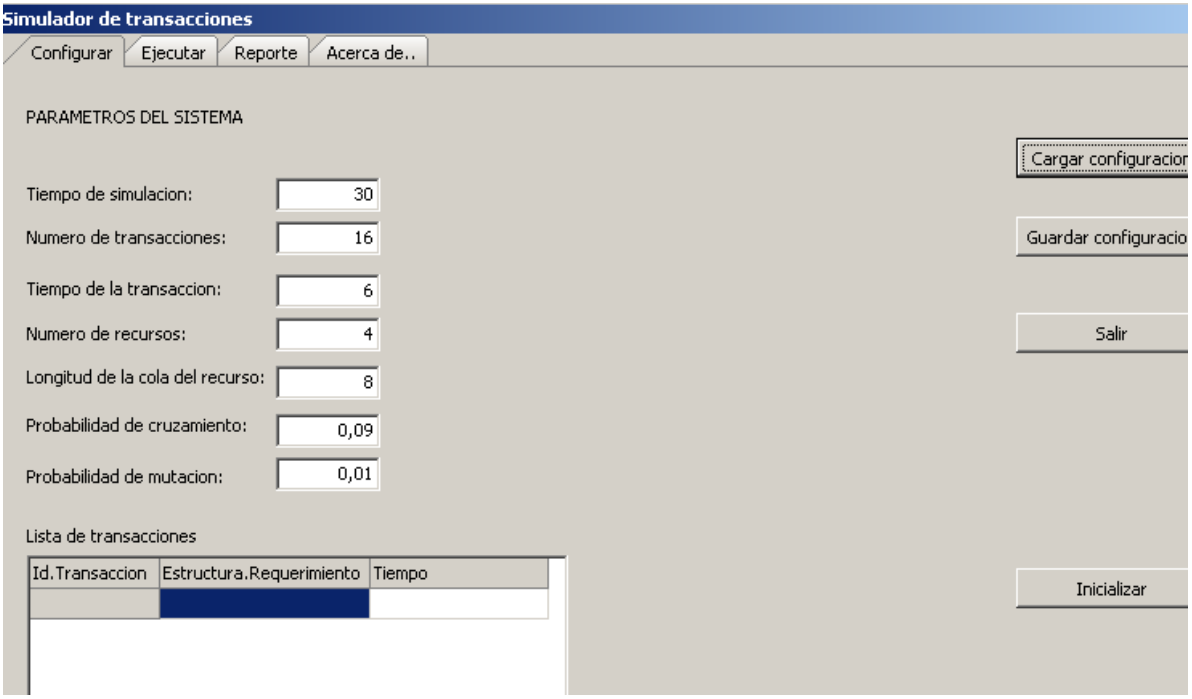


Figura 7.3. Ventana de parámetros del sistema.

De la figura 7.3, los parámetros del sistema que contiene los datos iniciales para ejecutar el simulador propuesto como:

Tiempo de simulación : 30 segundos.
Número de transacciones : 16.
Tiempo de la transacción : 6.
Número de recursos : 4.
Longitud de la cola del recurso: 8.
Probabilidad de cruzamiento : 0,09.
Probabilidad de mutación : 0,01.

7.3 INICIALIZAR

Ahora hacer clic en el botón **Inicializar** y aparecerá la ventanita de **Lista de transacciones** que contiene datos de: identificación de transacción, estructura de requerimiento y tiempo; R significa lectura y W significa escritura para sistema. Se muestra en la figura 7.4 lista de transacciones, siguiente:

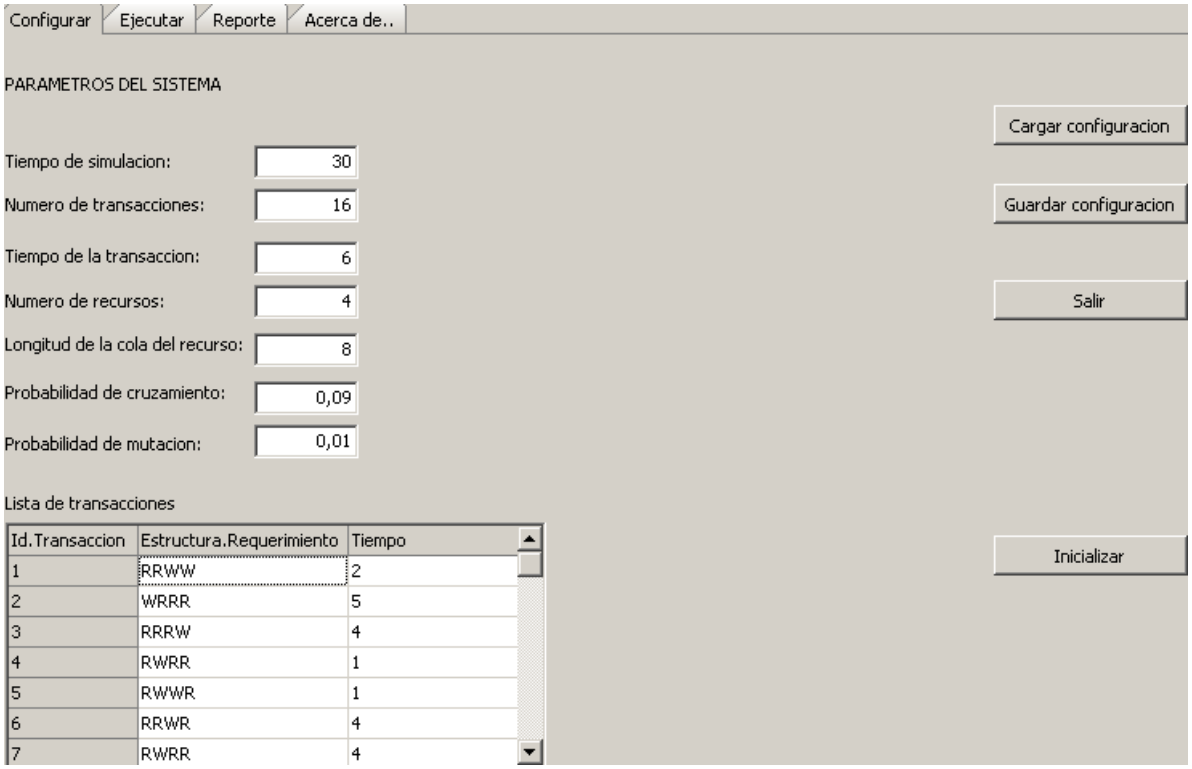


Figura 7.4. Lista de transacciones.

7.4 EJECUCIÓN

Pulsar la pestaña de **Ejecución** y aparecerá la ventana de ejecución, se muestran 3 bloques. El primer bloque es ‘parámetros del sistema’, este bloque muestra los datos iniciales para ejecutar la transacción.

El segundo bloque muestra las ‘colas de recursos’, es la administración de colas de transacción del simulador.

El tercer bloque muestra las “transacciones”, es la administración de ejecución de transacciones del simulador y se muestra en la figura 7.5 transacciones, siguiente:

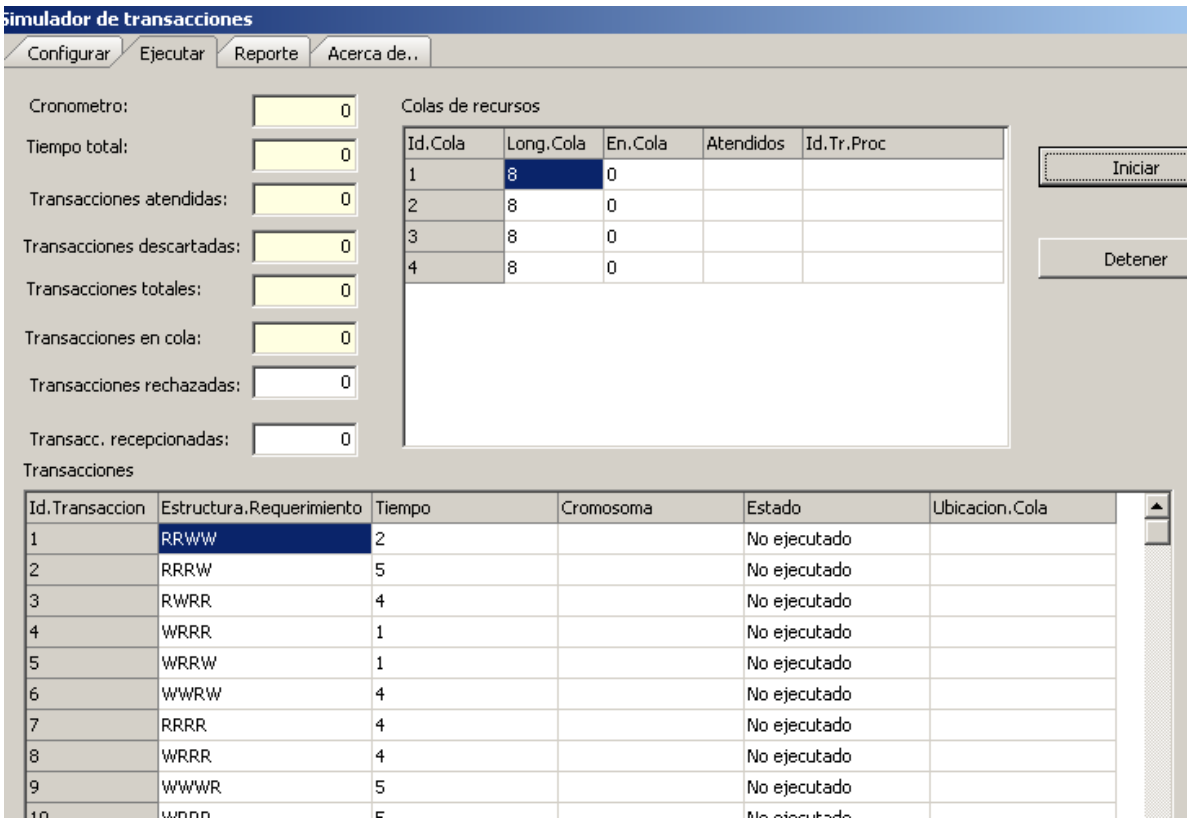


Figura 7.5. Transacciones.

Ahora, hacer clic sobre el botón **Iniciar** para ejecutar la transacción, los que se muestran en la figura 7.6 el proceso de transacciones y en la figura 7.7 la terminación del proceso de transacciones siguientes:

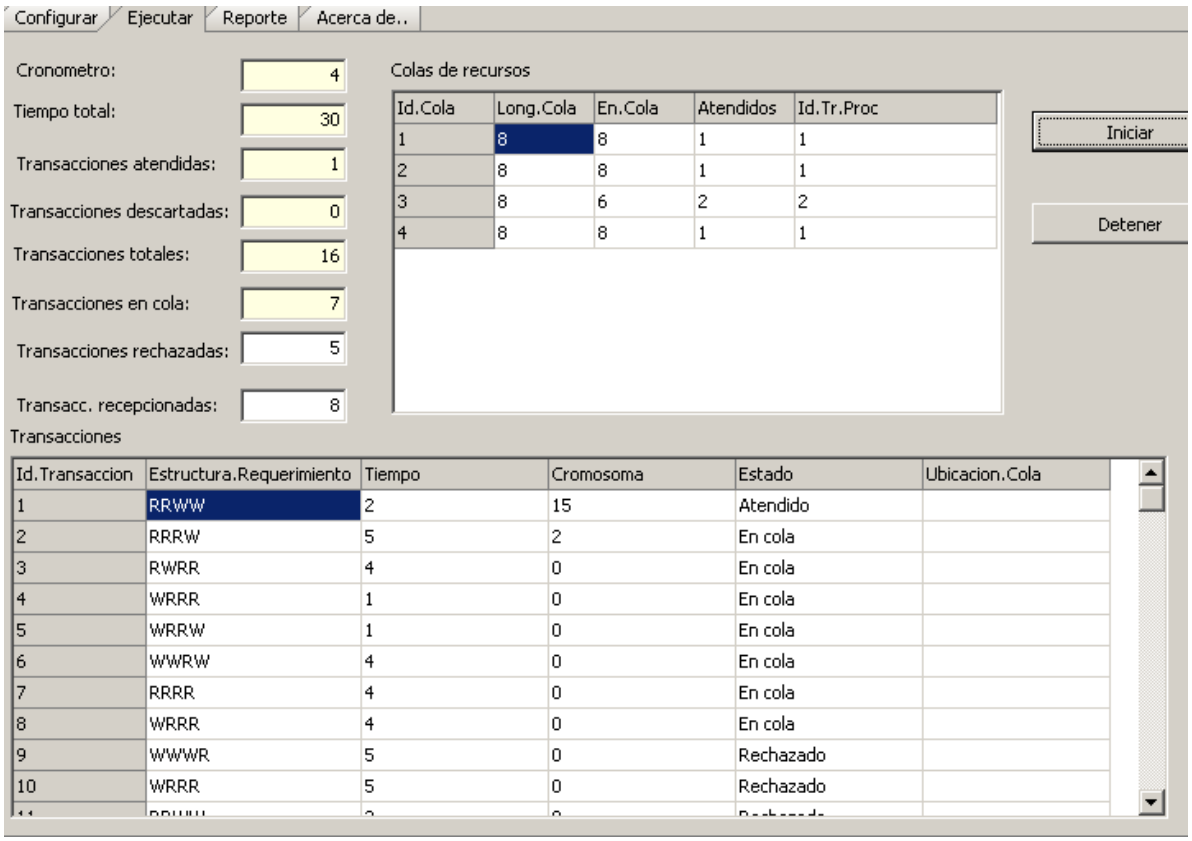


Figura 7.6. Proceso de transacciones.

Por ejemplo, en la ventanita de **parámetros del sistema,** están variando los datos iniciales, asimismo en las **colas de recursos** están variando los datos, en las **transacciones** están variando los datos y también en el **estado** están variando la cola atendida de cada uno.

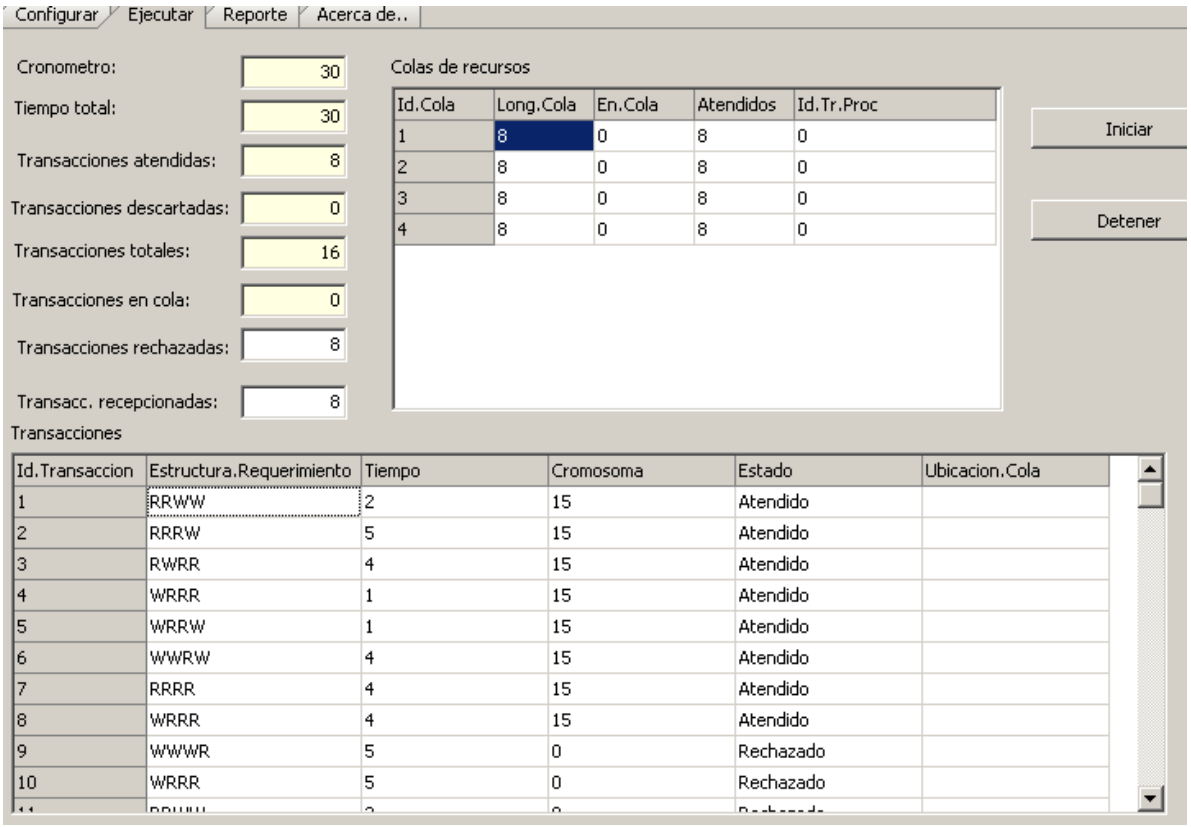


Figura 7.7.Terminación de procesos transacciones.

Por ejemplo, en **atendidos** todos tienen **8**, asimismo **en cromosomas** todos tienen 15 (1111 = 15), en **estado, 8 atendidos** y en rechazados **8** que no han cumplido las condiciones.

7.5 REPORTES

El sistema muestra en línea la evolución de la simulación. Está se envía a un reporte de salida. Se puede guardar el reporte en un archivo de texto. Seleccionando el botón **Guardar** para salvar reporte.

Se Pulsa la pestaña **Reportes** para presentar el resultado del simulador, que se muestra en la figura 7.8 reportes, mas adelante.

En reportes se ha considerado solo las transacciones procesadas como óptimos, para analizar los procesos efectuados por el sistema; los procesos rechazados son aquellos que no cumplen las condiciones necesarias de pares de cromosomas al momento de seleccionar de aptitudes fuertes, es decir, los de mejor salud.

.

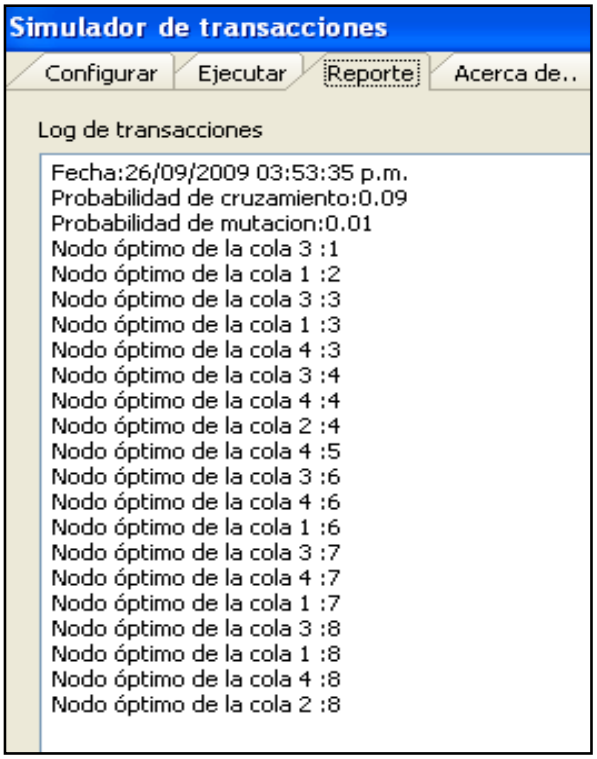


Figura 7.8. Reportes.

7.6 ACERCA DEL SISTEMA

Ahora, presionando la opción **Acerca del sistema**, aparecerá la ventana correspondiente, trata sobre versión y el autor del sistema aplicativo, y se muestra en la figura 7.9 acerca del sistema siguiente:

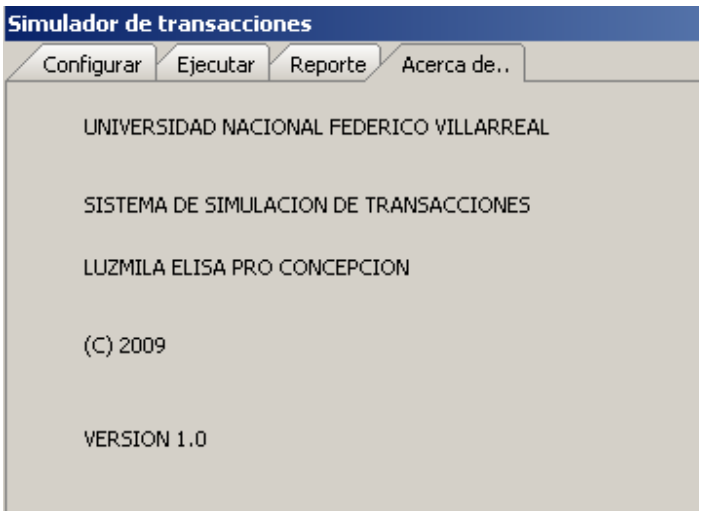


Figura 7.9. Acerca del sistema.

En conclusión, se ha implementado un “simulador prototipo de aplicación utilizando algoritmos genéticos para optimizar procesos de transacciones de datos”; para la tesis de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la Web”, que permitirá mejorar la asignación de transacciones de datos en el servidor, menor costo de recursos computacionales y el resultado devolverá a los usuarios / clientes en menor tiempo de procesos de transacciones, esté se realizará a través de despachador del sistema local y/o remoto. Dicho simulador se puede utilizar en los centros de cómputo de las empresas, instituciones, y centros de investigaciones.

El Modelo propuesto simulador de aplicación de algoritmos genéticos para optimizar procesos de transacciones en el servidor Web, el simulador asignará a los procesos de transacciones los siguientes: número población para la transacción, número de individuos, probabilidad de cruce, probabilidad de mutación, colas atendidas y no atendidas, tiempo consumido, resultados de transacciones óptimos. Luego, el tiempo de procesamiento de datos obtenido es menor que el tiempo de procesamiento de datos en un servidor Web convencional.

CONCLUSIONES

En el desarrollo de la investigación de la tesis de "Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la *Web*", ha sido posible llegar a las siguientes conclusiones:

1.- La evolución de Internet e Internet2 ha llegado a una alta proliferación y rendimiento en consultas de informaciones en el servidor *Web* por usuarios / clientes del mundo, que justamente están utilizado sistemas de base de datos y/o base de datos distribuidos. (De página. 23)

2. Hay deficiencia en el tiempo de procesos de transacción por el procesador del servidor; que actualmente trabajan con algoritmos tradicionales; como la lectura / escritura de datos en el disco magnético en el servidor Web, produciéndose por ejemplo, demora en la cola de espera, demora en tiempo de proceso de transacción, demora en tiempo de respuesta, que ocasionan los denominados cuellos de botella, falta memoria, etcétera. (De páginas 21 y 23)

3. El problema central que se propone está orientado al crecimiento y, evolución del servidor web de una manera económica y escalable que lleva a un rendimiento óptimo. Por consiguiente, existe la necesidad de estudiar los procesos de transacciones del sistema, de tal manera que se aplique otra alternativa como los algoritmos genéticos para optimizar el proceso de transacción en el servidor, a fin de mejorar los procesos del servidor web y, para mejorar la atención a los clientes / usuarios. (De página 23)
4. La estructura de un cromosoma consta de un grupo de alelos y cada uno corresponde con un recurso solicitado. El administrador de transacciones tomará el requerimiento por el recurso para ponerlo en cola, el que tenga un máximo de cromosoma igual a 1, es decir, cuando encuentre entre el grupo de transacciones, la transacción que tenga sus alelos en 1. Se tomará como cromosoma una cadena binaria que será convertida a números enteros.(De páginas 50, 58,59 y 125)
5. Se ha realizado un análisis de los modelos de transacciones que operan actualmente y se ha extraído tales mecanismos para llevarlo a un proceso de toma de decisiones en función de los algoritmos genéticos. (De pagina 125)
6. Se ha implementado un simulador prototipo para un sistema de aplicación con algoritmos genéticos, para optimizar el proceso de transacción, antes de procesar los datos, se evaluarán los procesos de transacciones sobre: tiempo de simulación, número de transacciones, tiempo de la transacción, número de recursos, longitud de la cola del recurso, probabilidad de cruzamiento, probabilidad de mutación, transacciones en cola, atendidos, en lectura, en escritura, tiempo consumido, y se consigue los resultados de procesos óptimos; el tiempo de procesamiento de datos mediante el simulador es menor que el tiempo de procesamiento de datos que en el procesador convencional, mejor uso del recurso de la computadora. (De pagina 148).

RECCOMENDACIONES

Del desarrollo de la investigación de la tesis de “Base de Datos Distribuidos Usando Algoritmos Genéticos Para Optimización de Proceso Transacción en la *Web*”, se puede recomendar realizar las investigaciones siguientes:

1. Base de Datos Distribuidos usando Algoritmos Genéticos para optimización de procesos de transacción en la Web, con solicitud de servicios concurrentes, que se permitan realizar operaciones de lectura /escritura, copias, consultas,

2. Base de Datos Distribuidos usando Algoritmos Genéticos para optimización de procesos concurrentes heterogéneos en servidor Web.

3. Base de Datos Distribuidos usando Redes Neuronales para Optimización de Transacción en el servidor Web.

4. Base de Datos Distribuidos para la Gestión de Transacciones en Servidor Web, para optimización con sistemas paralelos.

REFERENCIA BIBLIOGRÁFICA

1. ALMQUIST. P.
Type of service in the Internet protocol suite. Estados Unidos, Request for Comments (Proposed Standard) 1349, Internet Engineering Task Force, July 1992.
2. ÁLVAREZ MAURICIO, Octavio.
Sistemas Inteligentes Para Optimización de Eventos Concurrentes en la *Web*, Tesis Doctoral. Perú, Lima; Escuela Universitaria de Post Grado de la Universidad Nacional Federico Villarreal, 2006.
3. ARAUJO CÁRDENAS, Alfonso.
Sistemas Distribuidos, Conceptos y Características, 2004.
4. BUADES RUBIO, Gabriel.
Sistemas Distribuidos. Departamento de Investigación de Ingeniería Informática de la UIB, 2002.
5. CIBERTEC, EL COMERCIO Y PC WORLD.
Manual de Informática Práctica. Lima, Editorial el Comercio, 2000.
6. COULOURIS, George.
Sistemas Distribuidos. España, Madrid; Addison Wesley, 2001.
7. DARWIN, Charles Robert.
Teoría Biológica Sobre la Evolución. Inglaterra, investigación científica, 1930.
8. DAVIS, L.
Manual de Algoritmos Genéticos. Van Nostrand Reinhold, Nueva York, 1991.
9. DEITEL, H. M.
Introducción a los Sistemas Operativos. México, Addison-Wesley Iberoamericana, 1987.
10. DUEÑAS RODRÍGUEZ, Francisco Armando.
Sistemas Operativos Distribuidos. México, Dpto de Investigación de Ciencias de la Computación de la Universidad La Salle, malito:fduenas@hotmail.com, 2003.
11. HERNÁNDEZ SAMPIERI, Roberto; FERNÁNDEZ COLLADO, Carlos y BAPTISTA LUCIO, Pilar.
Metodología de la Investigación. México D. F., Editorial Esfuerzo S.A., 1999.
12. FOGARTY, C. A.
Variando la Probabilidad de Mutación del Algoritmo Genético. En [344], pp. 104-109, 1989.
13. GARCIA TOMÁS, Jesús; FERRANDO GIRÓN, Santiago y PIATTINI VELTHUIS, Mario.
Redes Para Proceso Distribuido. México, D. F., Editorial Rama, 1997.
14. GARCÍA CARBALLEIRA, Félix.
Sistemas distribuidos. Departamento de Investigación de Telemática, 1999.
15. GETTYS, J.
Problema de Sentencia HTTP.
<URL:http://www.w3.org/Protocols/HTTP-NG/951005Problem.html>, W3C, 1995.

16. GREFENSTTE, J. J.
Optimización de Parámetros de Control Para Algoritmos Genéticos, Sistemas Transacciones de IEEE, Hombre y Cibernética. Vol. 16, Ningún 1, el pp. 122-128,1986.
17. HANSEN, Gary y HANSEN, Janes.
Diseño y Administración de Bases de Datos. España. Editorial Prentice Hall, 1997.
18. HILERA GONZÁLEZ, J. R.; MARTÍNEZ HERNANDO, V. J.
Redes Neuronales Artificiales -Fundamentos, Modelos y Aplicaciones. Delaware- USA, Addison-Wesley Iberoamericana, 1995.
19. HOLLAND, John H.
Funciona el Camino Real, Compendio del Algoritmo Genético. Universidad de Michigan, vol. 7, número 22, 1975.
20. http://www.frontiertech.com/prodinfo.htm
21. http://www.fortunecity.es/sopa/chinchulines/812/informacion/noscs.htm (Sistemas Operativos).
22. http://dmi.uib.es/~bbuades/sistdistr/sld007.htm (Sistemas Distribuidos).
23. http://members.fortunecity.es/lrmdl/so7.htm#vsdrc (Sistemas Distribuidos).
24. http://sacbeob.8m.com/tutoriales/bddistribuidas/ (Base de Datos Distribuidas).
25. http://pdf.rincondelvago.com/bases-de-datos-distribuidas.html (Base de Datos Distribuidas)
26. http://www-lt.ls.fi.upm.es/sistemas_dist/Introduccion.pdf (Sistemas Distribui dos).
27. http://www.dia.eui.upm.es/cgi-in/asigfram.pl?cual=sis_dis&nombre=sistemas- dis tribu%eddos (Sistemas Distribuidos: Aplicaciones).
28. http://www.fisica.uson.mx/carlos/WebServices/WSOverview.htm (Objetos Distribuidos).
29. http://di002.edv.uniovi.es/~lourdes/publicaciones/bt99.pdf (objetos distribuidos).
30. ttp://www.inforg.uniovi.es/ia/Archivos/Apuntes%20y%20t/AlgoritmosGeneticos.pdf
31. HURTADO JARA, Omar.
Sistemas Distribuidos. España, Departamento de Investigación de la Universidad Carlos III de Madrid, OmarH@ie.inf.uc3m.es, 1997.
32. IGLESIAS, A.
Curso Avanzado de Métodos Numéricos. Argentina, Corrientes, 1998.
33. JANSSEN, B.
Proyecto PARC HTTP-NG. <URL: http://www.parc.xerox.com/-istl/projects/http-ng/>, 1997.
34. JOYANES AGUILAR, Luis.
Programación Orientada a Objetos. España, Mc Graw Hill/Interamericana, 1998.
35. JOYANES AGUILAR, Luis; ZAHONERO MARTÍNEZ, I.
Estructura de Datos-Algoritmos, Abstracción y Objetos. España, Mc Graw Hill/Interamericana, 1998.
36. KWAN, T. T., MCGRATH, R. E. y REED, D. A.

Modelos de Accesos de Usuarios al Servidor de Web de NCSA. Informe Técnico de UIUCDCS-R-95-1934, dpto. de Comp. Sci., Universidad de IL, febrero 1995.

37. MALPANI, R., LORCH J. y BERGER D.
Utilice Caché en los Servidores Cooperativos en Web. IV Conferencia Internacional de Web, páginas 107-117, Boston, diciembre de 1995.
38. MICHALEWICZ, Zbigniew.
Los Algoritmos Genéticos y Estructura de Datos con Evolución de Programas. Departamento de las Ciencias de la Computación de la Universidad de Carolina del Norte, Estados Unidos,1997.
39. MUÑOZ PÉREZ, Miguel Ángel.
Funcionamiento de un Algoritmo Genético. Abril, 2005.
40. PENROSE, R.
La Nueva Mente del Emperador. España, Mondadori España S. A., 1991.
41. PETERSON, J. L. y SILBERSCHATZ, A.
Operating Systems Concepts. MA-USA, Addison-Wesley, 1991.
42. PRESSMAN, Roger S.
Ingeniería de Software. España, Madrid; McGraw-Hill Interamericana, 2002.
43. SCHAFFER, J.,
Procedimientos de III Conferencia Internacional de los Algoritmos Genéticos. Publicadores de Morgan Kaufmann, San Mateo, CA, 1989.
44. SERVATI, Al y BREMNER, Lynn y LASI, Anthony.
La Biblia de Intranet. México, D. F., Editorial McGraw -Hill, 1998.
45. SILBERSCHATZ, Abrahm; KORTH, Henry y SUDARSHAN, S.
Fundamentos de Bases de Datos. España, quinta edición, McGrawHill, 2006.
46. STALLINGS, W.
Data and Computer Communications. NJ-USA, Prentice Hall, 1997.
47. TANENBAUM, A. S.
Operating Systems: Design And Implementation. NJ-USA, Prentice Hall, 1987.
48. TANENBAUM, A. S.
Organización de Computadoras-Un Enfoque Estructurado. México, Prentice Hall Hispanoamericana S. A., 1996.
49. TANENBAUM, A. S.
Sistemas Operativos Distribuidos. México, Prentice Hall Hispanoamericana, S. A., 1996.
50. TANENBAUM, A. S.
Redes de Computadoras. México, Prentice Hall Hispanoamericana S. A., 1997.
51. URUEÑA LEÓN, Edsel Enrique.
Direccionamiento IP. España, Departamento Investigación de Redes y Telecomunicaciones, edseleon@yahoo.com.mx, 2005.
52. WHITLEY, D.
El Algoritmo de Genitor y Presión de la Selección: Por qué la Línea de Asignación Basado de Ensayos Reproductores son Mejores. En los procedimientos

de la III Conferencia Internacional de los Algoritmos Genéticos. Páginas 116-121, San Mateo, CA, 1989.

53. ZURDO SAIZ, David; SICILIA BURGOA, Alejandro y ACEVEDO QUERO, Fernando.
Guía Rápida de Internet. España, Madrid; Paraninfo, 1997.
